
Universidade de Aveiro Departamento de Electrónica e Telecomunicações
2004

Pedro Alexandre Sousa Gonçalves

Implementação de um gestor de Qualidade de Serviço para redes heterogéneas IP

Universidade de Aveiro
2004

Departamento de Electrónica e Telecomunicações

Pedro Alexandre Sousa Gonçalves

Implementação de um gestor de Qualidade de Serviço para redes heterogéneas IP

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica do Professor Doutor Rui Luís Andrade de Aguiar, Professor Auxiliar do Departamento de Electrónica e Telecomunicações da Universidade de Aveiro

o júri

presidente

Prof. Dr. José Luis Guimarães Oliveira
professor associado da Universidade de Aveiro

Professora Doutora Teresa Maria Sá Ferreira Vazão Vasques
professora auxiliar do Instituto Superior Técnico da Universidade Técnica de Lisboa

Prof. Dr. Rui Luis Andrade de Aguiar
professor auxiliar da Universidade de Aveiro

agradecimentos

Agradeço em especial à minha esposa e aos meus filhos pelo apoio e pela paciência que tiveram durante o período em que desenvolvi a dissertação. Agradeço também ao meu orientador Professor Doutor Rui Aguiar, e ao Eng. Victor Marques. Um último agradecimento para os jovens engenheiros Diogo Gomes, Nuno Duarte, Nuno Sénica e João Paulo Barraca que mantiveram o demonstrador da rede e para a Drª Anabela Moreira.

resumo

O presente trabalho propõe um gestor de Qualidade de Serviço para redes heterogéneas IP.

O documento é composto por uma descrição das principais arquitecturas e tecnologias utilizadas neste tipo de redes, um levantamento das soluções de Bandwidth Brokers existentes e uma descrição da solução desenvolvida.

Começa-se por indicar as motivações gerais que levaram à execução deste trabalho e enumerar os seus objectivos. Sumariam-se os resultados do trabalho realizado e as publicações que dele resultaram.

Em seguida é descrito o conceito de Qualidade de Serviço e a sua importância na gestão de redes IP. São apresentados alguns modelos de Qualidade de Serviço como as arquitecturas *Integrated Services*, *Differentiated Services*, e *MultiProtocol Label Switching*, no que diz respeito às suas principais características, bem como a integração entre esses diferentes modelos.

Segue-se uma discussão dos mecanismos de controlo de Qualidade de Serviço, descrevem-se os mecanismos de controlo de acesso e de tráfego, os mecanismos de gestão de utilizadores (os protocolos RADIUS e DIAMETER) e de gestão de Qualidade de Serviço (protocolo COPS). São ainda analisadas as diferenças e as vantagens de uma política de uso de mecanismos de gestão distribuída vs. mecanismos de gestão centralizada.

Em seguida são levantados os requisitos para o desenvolvimento e implementação do Bandwidth Broker e é feito o levantamento e a análise das implementações existentes. Faz-se ainda a avaliação e comparação das implementações do Bandwidth Broker à altura do domínio público.

Depois faz-se a descrição da arquitectura do Bandwidth Broker implementado e dos interfaces dos elementos internos que o compõem. Faz-se ainda a descrição dos testes do sistema e análise dos resultados dos mesmos.

De seguida apresentam-se as conclusões finais e as dificuldades encontradas na execução do trabalho, com a indicação de potencial trabalho futuro a desenvolver nesta linha de trabalho. Verificou-se que apesar da escalabilidade do software desenvolvido se torna impossível que uma só máquina faça a gestão de uma rede de um operador.

É ainda apresentada no final uma lista de referências bibliográficas organizada por ordem da utilização no texto.

abstract

The present work suggests a Quality of Service Manager for heterogeneous IP networks.
This document is consisted of a description of the main architectures and technologies used in this kind of networks, a list of the existing Bandwidth Broker solutions and a description of the developed solution.
Firstly, there is a reference to the general motivations of this work, as well as a specification of its aims. The results and a list of publications that originated from it are then presented.
Then, it is described the concept of Quality of Service and its relevance to IP network management. Some models of Quality of Service are presented, such as Integrated Services, Differentiated Services and Multiprotocol Label Switching, as far as its main characteristics are concerned, as well as the integration between those different QoS models.
After that, there is a discussion of the QoS control mechanisms, access control and traffic control mechanisms. The user management (protocols RADIUS e DIAMETER) and QoS management mechanisms (protocol COPS) are also described. There is also an analysis of the differences and the advantages of distributed management versus centralized management broker use policies.
Next, there is a list of the requirements for the development and implementation of the Bandwidth Broker and an analysis of the existing implementations. There is also an evaluation of the Bandwidth Broker required functionality, as well as a comparison of the current public domain implementations.
After that there is a description of the implemented Bandwidth Broker architecture, its interfaces, and its internal components. There is still a description of the tests of the system and an analysis of the results.
The final conclusions come next followed by an enumeration of the difficulties found in doing such work. It was verified that, despite the scalability of the developed software, it is impossible for just one machine to be able to manage an entire operator network.
At the end, there is a bibliography organized by order of appearance in the text.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

LISTA DE ACRÓNIMOS

3G	Terceira geração
AAAC	*Authentication Authorisation Accounting and Charging*
AAAC.f	*Foreign AAAC*
AAAC.h	*Home AAAC*
AR	*Access Router*
ASAP	*As Soon As Possible*
AVP	*Attribute Value Pairs*
BB	*Bandwidth Broker*
BNF	*Backus Naur Form*
CAT	*Client Accept*
CC	*Client Close*
CdS	Classes de Serviço
CHAP	*Challenge Handshake Authentication Protocol*
CLI	*Command Line Interface*
CMS	*Cryptographic Message Syntax*
CN	*Correspondent Node*
CoA	*Care Of Address*
COPS	*Common Open Policy Service*
COPS-RSVP	*Common Open Policy Service - Resource Reservation Protocol*
COPS-PR	*Common Open Policy Service PRovisioning*
DiffServ	*Differentiated Services*
DRQ	*Delete Request State*
DRR	*Deficit Round-Robin*
DSCP	*Differentiated Service Code Point*
EAP	*Extended Authentication Protocol*
FHO	*Fast Handover*
FIFO	*First In First Out*
IETF	*Internet Engineering Task Force*
IntServ	*Integrated Services*
IP	*Internet Protocol*
LB	Largura de Banda
LDP	*Label Distribution Protocol*
LPDP	*Local Policy Decision Point*
LSP	*Label Distribution Protocol Peer*
LSR	*Label Switching Routers*
MN	*Mobile Node*
MPLS	*Multi-Protocol Label Switch*
MT	*Mobile Terminal*
MTU	*Maximum Transport Unit*
nAR	*New Access Router*
NAS	*Network Access Server*
NASREQ	*Network Access Server REQuirements*
NMS	*Network Management System*
NVUP	*Network View of User Profile*
oAR	*Old Access Router*
OPN	*Client Open*
OPWA	*One Pass With Advertisement*
PAP	*Password Authentication Protocol*

PDA	*Personal Digital Assistant*
PDP	*Policy Decision Point*
PEP	*Policy Enforcement Point*
PHB	*Per-Hop Behaviour*
PIB	*Policy Information Base*
POP	*Point of Presence*
PPP	*Point-to-Point*
PRC	*Provisioning Class*
PRI	*Provisioning Instance*
PRID	*Provisioning Instance Identifier*
QdS	Qualidade de Serviço
RFC	*Request For Comments*
ROAMOPS	*ROAMing OPerations*
RPT	*Report State*
RR	*Round-Robin*
RSVP	*Resource Reservation Protocol*
SLA	*Service Level Agreement*
SLS	*Service Level Specification*
SSC	*Synchronise Complete*
SSQ	*Synchronise State reQquest*
VOIP	*Voice Over IP*
WI-FI	*Wireless Fidelity*
WRR	*Weighted Round-Robin*
WWW	*World Wide Web*
GPS	*Generalised Processor Sharing*
WFQ	*Weighted Fair Queuing*
PGPS	*Packet Generalised Processor Sharing*
SCFO	*Self Clock Fair Queuing*
ASCII	*American Standard Code for Information Interchange*
DNS	*Domain Name Service*
RAPWG	*Admission Policy Working Group*

1. INTRODUÇÃO

1.1 ENQUADRAMENTO

O tráfego IP teve nos últimos seis anos um crescimento significativo. Com a massificação do World Wide Web (WWW) o tráfego Internet cresceu exponencialmente, tanto no número de servidores de conteúdos, como no número de utilizadores. As figuras 1 e 2 dão uma noção bastante precisa da amplitude desse crescimento ([1], [2]).

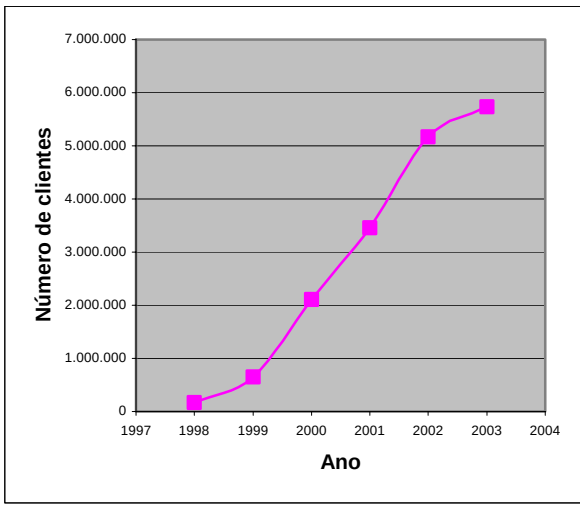

Figura 1 - Evolução do número de clientes do serviço de acesso à Internet em Portugal

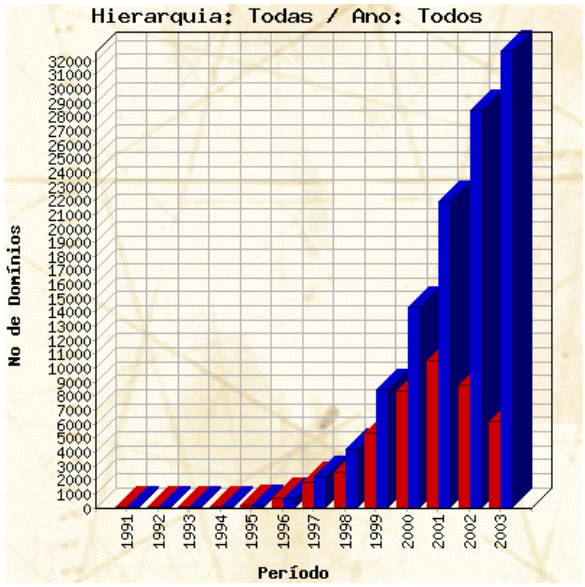

Figura 2 - Crescimento do número de domínios registados na FCCN

Os conteúdos foram sendo enriquecidos em termos gráficos, tendo-se vulgarizado o uso do vídeo, do som e as imagens. Actualmente existem sites onde se podem encontrar emissões de rádio e de televisão, emitidas via Web e acessíveis de todo o mundo. Com os requisitos de tráfego existentes neste tipo de aplicações e com a vulgarização da utilização das mesmas, a quantidade de informação que necessita de ser transportada cresceu de uma forma muito significativa. A Figura 3 mostra um gráfico que dá uma noção desse crescimento. Actualmente assiste-se a uma massificação da utilização do protocolo IP, com transporte de múltiplos tipos de tráfego através desse protocolo. Um exemplo de utilização do protocolo IP para o transporte de outra informação é o transporte de *voz sobre IP* (VOIP). Existem já grandes empresas que usam dispositivos de VOIP para todas as suas chamadas de voz, usando a infra-estrutura interna de comunicação de dados para

transportar voz. Um cenário previsto actualmente para o futuro próximo é a gradual utilização do IP para transporte de todo o tipo de informação, designado como *all over IP*.

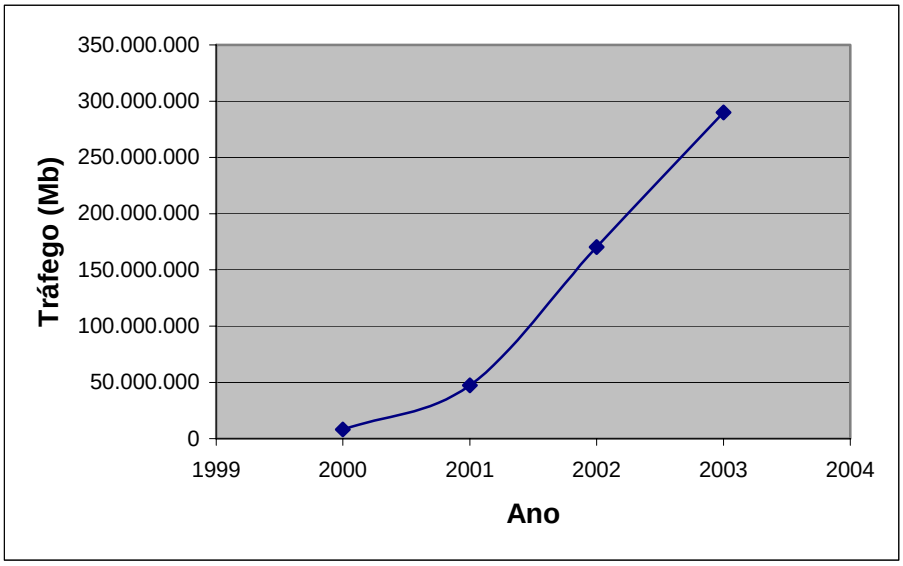

Figura 3 - Variação da quantidade de tráfego na comunicação de dados por pacotes

Por outro lado verifica-se já neste momento um grande crescimento da utilização dos dispositivos IEEE 802.11 (Wi-Fi), com a instalação de equipamentos que permitem o acesso à Web a partir de PDA's e computadores portáteis nos maiores centros comerciais, nos aeroportos, nos hotéis e até no centro de algumas cidades. Poderemos daqui a algum tempo ter a atribuição de um endereço IP ao aparelho de ar condicionado, ao vídeo-porteiro, ao alarme ou micro-ondas das habitações. Estas novas utilizações estender-se-iam aos automóveis e a tudo o resto, correspondendo às necessidades de interligar todos os dispositivos com os quais houvesse interesse de interagir.

Esta tendência criaria uma diversidade de informação enorme, pelo que se torna um grande desafio encaminhar toda a informação para os respectivos destinos com as condições que cada tipo de tráfego exige. Os recursos como a largura de banda dos links e a capacidade de processamento dos routers são limitados. Com um aumento tão acentuado de tráfego, poderão ficar congestionados aumentando tanto a quantidade de pacotes perdidos, como o atraso na entrega de pacotes. Isto afecta muitas aplicações, que são muito sensíveis quer ao atraso quer à perda de pacotes, como por exemplo as aplicações de VOIP ou streaming de vídeo.

Para resolver o problema da degradação do atraso e da perda de pacotes existem duas alternativas: ou se aumentam as capacidades de transporte dos links e da capacidade de processamento dos elementos de rede; ou se estabelecem prioridades e se classificam as fontes de tráfego de acordo com a prioridade com que elas necessitam de ser encaminhadas, protegendo a

informação essencial. A primeira representa um investimento em infra-estruturas que não é viável para os operadores de telecomunicações. A segunda permite que o tráfego seja discriminado de acordo com o seu grau de prioridade, por forma a que se minimizem os atrasos e as perdas de pacotes de diferentes grupos de fluxos, e se tenham garantias de que uma chamada de VOIP ou que a recepção de um vídeo poderão chegar em perfeitas condições. Esta solução tem ainda a vantagem de permitir atribuir preços diferentes à utilização dos diferentes serviços, permitindo assim que os operadores de telecomunicações possam mais facilmente rentabilizar os investimentos feitos na melhoria das infra-estruturas de rede. Esta prioritização do tráfego é consequência da aplicação destas políticas de Qualidade de Serviço (QdS) ao encaminhamento de tráfego numa rede. Existem várias possibilidades de dotar as redes de mecanismos de controlo de admissão, entre as quais a integração de diferentes arquitecturas de QdS descrita no capítulo 2.1.4, ou então fazendo uso de uma entidade central que define as políticas da rede que entre outras responsabilidades tem a missão de fazer controlo de admissão da rede de acesso. Essa entidade denomina-se gestor de QdS, Bandwidth Broker (BB) ou QoS Broker.

Uma característica que a rede deverá possuir num futuro próximo será suporte de mobilidade: deverá suportar mobilidade geográfica dos utilizadores, permitindo-lhes que se possam deslocar geograficamente sem que percam as suas ligações à rede; e suportar mobilidade entre tecnologias, permitindo que ao chegar a um local onde haja oferta de várias tecnologias o terminal do utilizador possa comutar para a tecnologia de acesso que lhe permita um acesso mais barato ou com melhor oferta de recursos. Existem já alguns equipamentos com suporte para várias tecnologias de acesso, como computadores portáteis com acesso ethernet, Wi-Fi e com possibilidade de utilização de serviços de GPRS; PDA's com acesso bluethoth, WI-FI e possibilidade de ligação a ethernet, terminais GSM com suporte bluethoth e WI-FI. Falta que o suporte de mobilidade seja agora desenvolvido por parte da rede. Questões como as alterações necessárias às redes actuais no que respeita à tarifação e ao processo de *hand-over* precisam de ser estudadas de modo a que a mobilidade seja uma realidade comercial.

Este trabalho enquadra-se neste contexto, apresentando um gestor de Qualidade de Serviço (BB) para estes ambientes. Estes ambientes de mobilidade foram desenvolvidos dentro do projecto europeu Moby Dick [51], que proporcionou um conjunto de requisitos complexos para o projecto e desenvolvimento de um BB.

O gestor de QdS implementado neste trabalho faz a gestão de uma rede IP heterogénea que agrega tráfego de múltiplas tecnologias: tráfego ethernet, WI-FI e TD-CDMA. Dispõe ainda de suporte de mobilidade entre redes e entre tecnologias e encontra-se associado a entidades de facturação.

1.2 OBJECTIVOS

Os objectivos do trabalho desenvolvido foram:

- Estudar aspectos relevantes para gestor de QdS:
  - Proceder ao levantamento das implementações de BB existentes;
  - Fazer o levantamento dos requisitos existentes para o projecto do BB;
  - Estudar o protocolo COPS, bem como o formato das suas mensagens;
- Definir a arquitectura de um novo gestor de Qualidade de Serviço: definir quais os interfaces e comunicação com elementos exteriores e quais as tecnologias a usar;
- Proceder à implementação do Bandwidth Broker: implementar o sistema e criar bases de dados;
- Verificar a possibilidade de controlo de uma rede de acesso heterogénea com suporte de QdS, através de um servidor de políticas;
- Investigar a possibilidade de controlo conjunto de elementos de rede *IntServ* através da utilização de COPS-RSVP e da integração entre as redes com arquitecturas *IntServ* e *DiffServ*;
- Testar o sistema implementado: verificar o funcionamento de acordo com a arquitectura definida e proceder à medição dos tempos de resposta.

1.3 RESULTADOS DA DISSERTAÇÃO

Os resultados mais importantes do trabalho efectuado podem sumariar-se em:

- Demonstração da possibilidade de controlo de uma rede de acesso heterogénea com suporte de QdS;
- Demonstração da possibilidade de controlo conjunto de elementos de rede *IntServ* através da utilização de COPS-RSVP;
- Desenvolvimento de um BB para o demonstrador da rede, utilizado e testado no projecto Moby Dick [51].

Deste trabalho resultou ainda a publicação de dois artigos:

- Victor Marques, Carlos Parada, Pedro Gonçalves, Rui L. Aguiar, Francisco Fontes, "Next Generation Network Provider Architecture Demonstrator", Conftele2003 4th Conference on Telecommunications, Aveiro, Junho de 2003;

- Pedro Gonçalves, Diogo Gomes, Victor Marques, Rui L. Aguiar, "QoS Control support for heterogeneous networks", Conferência de Redes e Computadores 2003, Bragança, Setembro de 2003.

1.4 ORGANIZAÇÃO DA DISSERTAÇÃO

O primeiro capítulo da dissertação - Introdução - indica as motivações gerais que levaram à execução deste trabalho e enumera os seus objectivos. Apresenta os resultados do trabalho realizado e as publicações que dele resultaram.

No capítulo 2 - Qualidade de Serviço é discutido o conceito de Qualidade de Serviço e a sua importância na gestão de redes IP. São apresentados alguns modelos de QdS como o *IntServ*, *DiffServ*, MPLS. São discutidas as suas principais características e a possível integração entre diferentes modelos.

No capítulo 3 - Controlo do QdS, são discutidos mecanismos de controlo de QdS e discutidos os mecanismos de controlo de acesso e de tráfego. São ainda apresentados mecanismos de gestão de utilizadores e protocolos associados, como RADIUS e DIAMETER. Também são tratados problemas de gestão de QdS e protocolos associados, como o protocolo COPS. Finalmente é apresentada uma comparação entre políticas de uso de Brokers distribuídos vs. Brokers centralizados.

No capítulo 4 - Gestores de QdS são apresentados os requisitos para o desenvolvimento de Bandwidth Brokers e é feita uma análise das várias implementações existentes. O capítulo termina com a avaliação e comparação de implementações existentes em domínio público.

O capítulo 5 - Gestor de QdS com suporte de mobilidade contém a descrição da arquitectura do Bandwidth Broker implementado, dos interfaces dos elementos internos que o compõem. O capítulo termina com a descrição dos testes do sistema e a análise dos resultados dos mesmos.

No último capítulo - Conclusões apresentam-se as conclusões finais e discutem-se algumas dificuldades encontradas na execução do trabalho, com a indicação de algum trabalho futuro que pode ser desenvolvido nesta linha.

É ainda apresentada no final uma lista de referências bibliográficas organizada por ordem da utilização no texto.

2. QUALIDADE DE SERVIÇO

Assiste-se actualmente a uma massificação da utilização do protocolo IP para transporte de múltiplos tipos de tráfego, como transferência de dados, transporte de voz ou streaming de vídeo ou de áudio. Esses tipos tão diversos de tráfego têm também requisitos de largura de banda muito diversos, atraso de chegada e perda dos pacotes. Por exemplo uma aplicação de transferência de dados será muito tolerante a um atraso na chegada dos pacotes, mas uma aplicação de VOIP não o será.

O aumento de tráfego produzido pelo crescente número de utilizadores e de servidores cria também novos problemas: as capacidades das ligações são limitadas e com aumento de tráfego tendem a ficar saturadas. A saturação das ligações e dos *routers* faz aumentar os atrasos e aumenta as probabilidades de que os pacotes sejam descartados numa das filas dos routers.

Com vista à resolução dos problemas criados pelo aumento de tráfego numa rede que só suporta um tipo de serviço e que trata todas as fontes de tráfego do mesmo modo, o IETF criou um grupo de trabalho que definiu um modelo de QdS chamado Serviços Integrados (*IntServ*). Este modelo de QdS, que se descreve no capítulo 2.1.1, apesar de fornecer garantias de QdS aos fluxos admitidos para todo o percurso entre a origem e o destino, tem um sério problema de escalabilidade, o que o torna pouco adequado para redes de core [13].

Foi então criado um novo grupo de trabalho pelo IETF, com vista a criar um novo modelo de QdS que não sofresse dos problemas de escalabilidade, a que se chamou Serviços Diferenciados (*DiffServ*), descrito no capítulo 2.1.2. Este modelo foi melhor recebido pelos operadores e já não sofre dos problemas de escalabilidade do modelo anterior. Mas para fornecer garantia de QdS entre extremos da rede, necessita de mecanismos de controlo de admissão. Quando as condições de sobrecarga dos elementos de rede não permitam o serviço de um novo fluxo, este não deverá ser admitido, mas os fluxos anteriormente admitidos continuarão a ser servidos de acordo com os parâmetros de QdS a eles associados.

A definição de QdS pode ser feita em termos tão abstractos como "conjunto de parâmetros do serviço que garante a satisfação do utilizador". No âmbito de uma rede IP, os parâmetros de QdS que normalmente se pretendem controlar são fundamentalmente a largura de banda, o atraso e a sua variância.

Nas secções seguintes iremos ver a descrição de diferentes modelos de QdS e o estudo da interligação entre estes modelos.

2.1 MODELOS DE QDS

2.1.1 Serviços Integrados - *IntServ*

Em resposta à procura crescente de mecanismos de QdS, a *Internet Engineering Task Force* (IETF) criou o *Integrated Services (IntServ) Working Group*. Esta arquitectura foi desenvolvida para optimizar a utilização dos recursos de rede para aplicações que requerem limites do tempo de atraso e para as quais o tempo de entrega dos pacotes tem uma importância crítica, como aplicações multimédia em tempo real. Este tipo de aplicações não se adequa com o tráfego *best-effort* devido aos atrasos no encaminhamento e às perdas de informação quando ocorre congestionamento na rede. Na arquitectura *IntServ* [3] cada pacote é associado a um fluxo definido pelos seus endereços de origem e de destino e pelo número do porto.

2.1.1.1 Serviços Definidos

O *IntServ* permite três tipos de serviços distintos, a seguir descritos: *best-effort, guaranteed service e controlled load.*

2.1.1.1.1 Best-effort ou ASAP

O serviço *best-effort* (ou ASAP como também é conhecido) é usado por tráfego sem qualquer garantia de QdS. Neste tipo de serviço os pacotes são entregues logo que seja possível, sem qualquer garantia. O atraso dos pacotes é indeterminado à partida, pois não são reservados quaisquer recursos e por isso o atraso estará dependente da sobrecarga da rede naquele instante. Também não oferece garantia de entrega dos pacotes, uma vez que em caso de sobrecarga os *routers* fazem descarte dos pacotes, começando pelos pacotes de *best-effort*.

Num cenário de pouca sobrecarga da rede estes pacotes são entregues no destino sem perdas e com pequeno atraso. Quando a sobrecarga dos elementos da rede aumenta, os pacotes passam a ser entregues com atrasos maiores, podendo ser descartados por um dos elementos da rede, se a quantidade de pacotes a transmitir for muito elevada.

As aplicações mais indicadas para este tipo de serviço são aquelas em que o atraso na chegada dos pacotes é tolerado, de que são exemplos Telnet, FTP, HTTP, etc.

2.1.1.1.2 Guaranteed Service

O Serviço *Guaranteed Service* [4] (serviço garantido) fornece uma largura de banda entre extremos garantida sem perda de pacotes para fluxos que respeitem o perfil de QdS, sendo semelhante ao de uma linha comutada virtual. A sua grande vantagem é que é possível definir um atraso máximo para os pacotes na rede, não existindo descarte nas filas dos *routers* da rede.

É especialmente concebido para aplicações com requisitos específicos de largura de banda com intolerância, quer à degradação do nível de largura de banda disponível, quer à perda de pacotes.

As características do tráfego são sujeitas a análise, de modo a verificar a concordância com as condições definidas; O tráfego não conforme é tratado como *best-effort*, mas podendo acontecer que seja descartado.

2.1.1.1.3 Controlled Load

O serviço *controlled load* [5] (ou *predictive service* como também é conhecido) não apresenta garantias quantitativas, ao contrário do serviço *guaranteed service*. A garantia que este tipo de serviço apresenta é a de que o tráfego deste tipo de serviço não sofrerá degradação significativa com o congestionamento, como o que acontece para tráfego *best-effort*. Em termos de comportamento é semelhante ao serviço *best-effort* em redes pouco congestionadas. Os routers que implementem este serviço devem verificar se as características do tráfego estão de acordo com as condições definidas, de modo a tratar o tráfego não conforme como *best-effort* ou até para o descartar. Assim, este serviço é vocacionado para aplicações que tolerem alguma degradação da largura de banda e ocorrência de atraso, como aplicações de tempo-real que sejam adaptativas.

2.1.1.2 Componentes de um elemento de rede *IntServ*

Um elemento de rede *IntServ* pode ser dividido em várias entidades funcionais. A Figura 4 ilustra a constituição de um nó de serviços integrados. Os elementos internos podem ser identificados como:

- **Classificador de pacotes:** identifica e selecciona o tipo de serviço de cada pacote;
- **Escalonador:** selecciona os pacotes das diversas filas de espera, de modo a que estes sejam servidos cumprindo os requisitos de QdS requisitada;

- **Controlador de admissão**: determina quais os pedidos de reserva que podem ser aceites, de modo a que esse serviço possa ser concedido sem pôr em causa o serviço dos restantes fluxos já autorizados;
- **Protocolo de sinalização**: processa as mensagens recebidas e entrega-as ao controlador de admissão;
- **Processo de routing**: processo de routing existente no router que determina qual o próximo nó para o fluxo em questão;
- **Política de Controlo**: define um conjunto de políticas que determinam o comportamento do elemento de rede;

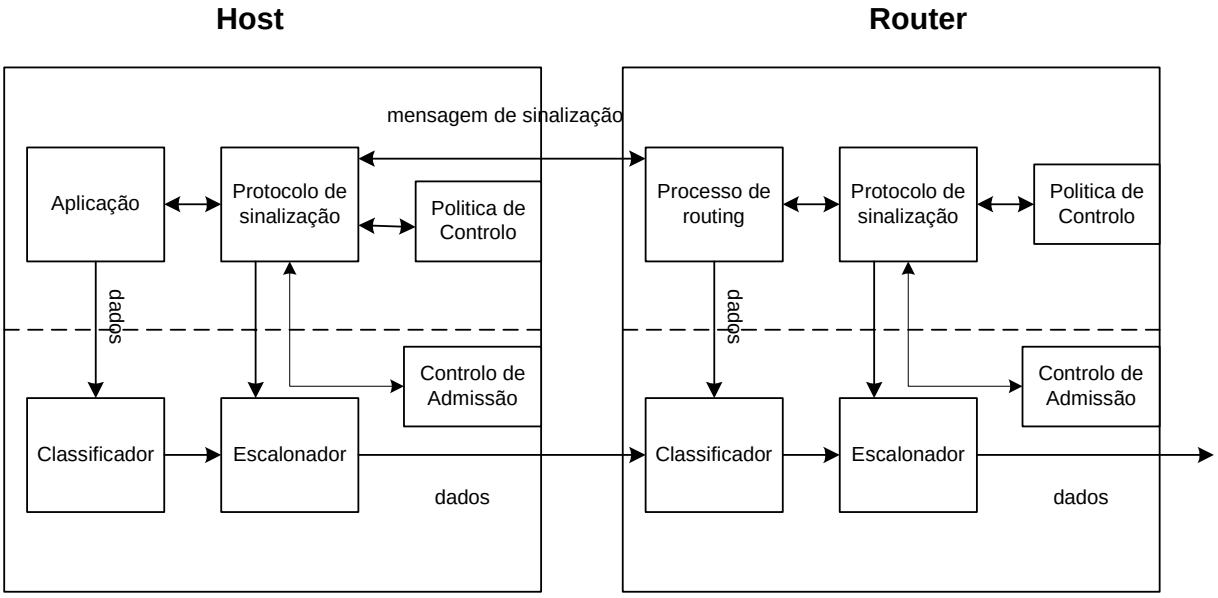

Figura 4 - Modelo de Serviços Integrados

Como se pode deduzir da Figura 4, a arquitectura *IntServ* supõe que possa existir mais do que um elemento de rede entre o originador e o destinatário do fluxo e que existem dois fluxos independentes: um para a sinalização que a arquitectura *IntServ* necessita, e outro de informação a ser transmitido entre o originador e o destinatário. Na figura estes dois fluxos estão separados pelas linhas a tracejado.

Para que os pacotes de um fluxo possam ser tratados com uma determinada qualidade de serviço é necessário que estes sejam identificados. Depois da identificação dos pacotes, estes são mapeados para uma classe de serviço, sendo encaminhados para a respectiva fila. A identificação

da classe e o seu mapeamento para a classe de tráfego a utilizar são feitos pelo classificador de pacotes.

Sendo uma classe uma abstracção de um router específico, um pacote pode ser mapeado em diferentes routers para diferentes classes. No classificador de pacotes o importante é garantir que a classe onde o pacote foi mapeado tenha um tratamento tal que o pacote obtenha a qualidade de serviço que necessita.

Normalmente o classificador usa a informação proveniente do cabeçalho IP do pacote: endereços de origem, de destino e do cabeçalho do protocolo da camada acima: protocolo e número de porto. É ainda possível que o classificador procure informação nas camadas acima, no cabeçalho da camada de aplicação. Esta implementação é complexa, pois requer a análise dos cabeçalhos dos pacotes de várias camadas protocolares, o que é muito dispendioso em termos de tempo.

O escalonador de pacotes é a entidade que gere a entrada em serviço dos diversos pacotes das várias filas do router, de acordo com a classe a que cada fila pertence. Tem que assegurar que o serviço de cada pacote está de acordo com os parâmetros de QdS definidos. Existem vários algoritmos de escalonamento que podem ser usados pelo escalonador, como prioritização estrita, Round-Robin, Fair-Queuing, Weight-Fair-Queueing, etc [8].

A qualidade de serviço em *IntServ* assenta no princípio dos routers se comprometerem a reservar os recursos para que os fluxos os possam usar. Para que esse princípio seja respeitado, é necessário que os recursos sejam previamente requisitados e que o router os reserve para esse uso. Para que o router se comprometa a fazer a reserva, é necessário que este possa fazer uma análise do seu estado de ocupação e da disponibilidade de um recurso ser disponibilizado. O pedido deverá ser recusado se não existirem recursos e qualquer admissão só pode ser feita se esse serviço puder ser prestado sem prejudicar o serviço dos fluxos que entretanto já tenham sido admitidos.

Cabe ao classificador policiar as propriedades dos fluxos e determinar se os pacotes de um fluxo estão de acordo com a reserva efectuada e se a largura de banda do mesmo não está a ser excedida. No caso de a política escolhida para aplicar aos pacotes que não respeitem as condições da reserva que foi feita ser o descarte, cabe ao escalonador executar esta acção.

2.1.1.3 Resource Reservation Protocol - RSVP

O Resource reSerVation Protocol [7] (RSVP) foi projectado para permitir aos elementos da rede (origem e destino de tráfego, e routers de passagem) a comunicação dos requisitos do tráfego e disponibilidade dos recursos, de modo a que esses elementos de rede se possam configurar para poderem prestar os serviços atrás descritos.

O RSVP identifica uma sessão através dos endereços origem e destino, do tipo de protocolo da camada de rede e do número do porto da máquina de destino. Cada sessão RSVP apenas se

aplica a pacotes de um determinado fluxo, por isso as características do mesmo têm que ser descritas nas mensagens RSVP.

O RSVP não é um protocolo de encaminhamento; É usado somente para reservar recursos nas máquinas ao longo da rota já definida à partida para o fluxo pelo protocolo de encaminhamento existente.

A Figura 5 ilustra o processo de propagação de mensagens de um fluxo, com a respectiva sequência.

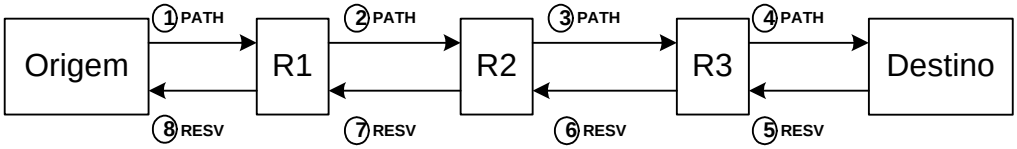

Figura 5 - Propagação das mensagens numa rede *IntServ*

O processo de reserva de recursos inicia-se pelo envio de uma mensagem PATH, por parte do originador do fluxo ao primeiro router da rota definida. Essa mensagem PATH contém, entre outra informação:

- **Phop:** informação do endereço do último nó da rede que contém facilidades RSVP;
- **Sender Template:** descrição do emissor da mensagem, contém o endereço IP do emissor e o porto;
- **Sender TSPEC:** define as características de tráfego do emissor;
- **ADSPEC**: trata-se de um campo opcional que é gerado pelos nós da rede e que descreve um conjunto de características dos elementos do percurso da rede, como a disponibilidade de recursos ou outros parâmetros necessários para que os mecanismos de controlo de QdS funcionem correctamente.

Cada mensagem PATH enviada pelo originador do tráfego é respondida por uma mensagem RESV que segue o caminho inverso da mensagem PATH, nó a nó. A mensagem PATH propaga por todos os nós do troço de rede entre a origem e o destino a informação das características de tráfego que o emissor pretende enviar. Assim, as mensagens PATH fornecem a informação respeitante aos requisitos de recursos de cada fluxo para todas as máquinas, desde a origem até ao destino.

Analisando o exemplo da Figura 5, verificamos que o originador envia uma mensagem PATH para o router R1. Após a analise da validade da mensagem R1, actualiza o campo *Sender Template* da mensagem recebida no que respeita à inserção do elemento *Phop*, de modo a que o

próximo elemento de rede saiba para onde deve enviar a mensagem RESV. Finalmente envia a mensagem que irá ser capturada pelo próximo elemento de rede *IntServ*. Os elementos de rede que se seguem repetem as mesmas acções até que a mensagem chegue ao destino.

A partir do destino do fluxo é iniciado o processo de resposta. Uma mensagem RESV faz a reserva de recursos, é enviada pelo mesmo caminho que foi usado pelas mensagens de PATH, mas agora no sentido inverso. Essa mensagem é propagada pelos vários elementos de rede presentes no percurso. Os routers que não tenham suporte RSVP não processam as mensagens e limitam-se a conduzi-las para o próximo elemento de rede.

Ao receber uma mensagem RESV, um router analisa o pedido através das suas rotinas de controlo de admissão: no caso de existirem recursos efectua a reserva dos mesmos para que possa ser possível prestar o serviço e envia a mensagem RESV em direcção à origem do tráfego, para que nesse troço da rede também se façam as reservas necessárias; caso não seja possível aceder a esse pedido de reserva, é enviada uma mensagem em direcção ao originador da mensagem RESV.

2.1.1.3.1 Utilização de OPWA para fazer uma reserva

A mensagem de PATH de uma sessão RSVP não contém em si qualquer informação para fornecer ao receptor sobre o estado da rede onde as mensagens de PATH passaram. Contém somente informação relativa ao serviço pedido pelo originador e informação acerca do último elemento de rede RSVP por onde passou.

Existe um melhoramento à arquitectura chamado *One Pass With Advertisement* (OPWA) que permite contornar esta limitação. Consiste em acrescentar à mensagem PATH um novo objecto designado ADSPEC, utilizado para colher informação acerca da disponibilidade de recursos da rede por onde a mensagem PATH passa. Inicialmente, esta informação é gerada pelo emissor do tráfego, descrevendo a previsão dos recursos que a aplicação vai necessitar. Ao longo da viagem das mensagens PATH, essa informação vai sendo actualizada pelos routers que interpretam RSVP. A mensagem PATH chega ao destino do fluxo, contendo a informação acerca da rede por onde vai passar o fluxo. O objecto ADSPEC contém, entre outros, os seguintes campos:

- Largura de banda máxima disponível ao longo de todo o caminho da rede;
- O tamanho do MTU máximo disponível ao longo de toda a rede;
- Informação acerca da existência de algum nó da rede que não suporte RSVP ao longo do caminho entre o originador e o destinatário;
- Informação acerca da implementação de um determinado serviço *IntServ* ao longo do percurso.

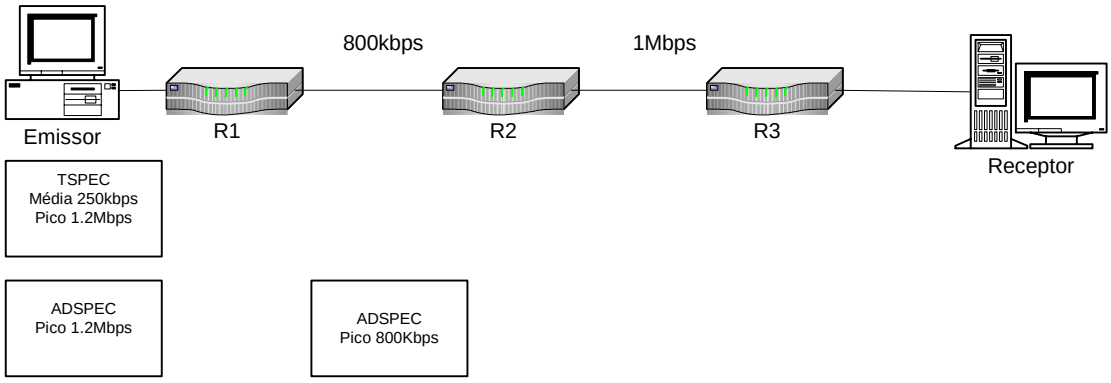

Figura 6 - Cenário de utilização de OPWA no RSVP

A Figura 6 ilustra um cenário de utilização do OPWA onde, se não fosse actualizado o ADSPEC pelos elementos da rede o pedido seria recusado pelo router R2, mas, devido ao facto de o receptor ter tido conhecimento das limitações da rede, fez uma reserva com uma largura de banda de pico mais baixo que assim pode ser aceite a reserva por todos os elementos da rede.

Essa informação pode permitir decidir qual o melhor tipo de reserva que deve ser feita com vista a transportar o fluxo, mas também com mais possibilidades de ser aceite.

2.1.1.3.2 Outras considerações acerca de RSVP

A lista que se segue detalha outros pormenores acerca do protocolo RSVP:

- As reservas de RSVP são unidireccionais. Quer isto dizer que no cenário ilustrado na Figura 6 seriam precisos dois tipos de reservas para um fluxo que se deslocasse do emissor para o receptor e para um fluxo que passasse em sentido contrário;
- O RSVP permite alterar as condições de uma reserva, como por exemplo, alterar a largura de banda ou outros parâmetros de QdS de uma reserva;
- Existem mensagens para fazer remoção de uma reserva, o que deve ser feito assim que uma aplicação deixe de necessitar dela;
- A validade das reservas RSVP é limitada. Para evitar que estas caduquem é necessário que elas sejam renovadas de tempos a tempos.

2.1.1.4 Comentários acerca da Arquitectura

A arquitectura *IntServ* tem sérios problemas de escalabilidade, o que se traduziu num sério revés na sua massificação. Estes problemas provêm da necessidade de manutenção do estado de todas as reservas em todos os nós do percurso. A manutenção do estado de todas as reservas de todos os fluxos em todos os nós da rede cria dois problemas graves:

- quantidades enormes de informação a armazenar nas memórias dos routers;
- tempos de acesso à informação de uma dada reserva muito grandes.

2.1.2 *DiffServ*

Numa tentativa de colmatar as dificuldades de escalabilidade existentes na arquitectura *IntServ*, o IETF criou em meados de 1997 *Differentiated Services Working Group* [9]. Este grupo definiu novos comportamentos de encaminhamento dos routers (*hop*), chamados *per-hop behaviors* (PHB). Estes comportamentos descrevem a influência do router num fluxo de pacotes, o que se traduz no serviço disponibilizado a um conjunto de fluxos com o mesmo serviço, processando agregados de informação.

Foi definido um número limitado de serviços a fornecer pela rede e uma arquitectura simples para os nós do core, de modo a alcançar os objectivos de escalabilidade da rede, mantendo a diferenciação de tratamento adequada, baseada na marcação existente nos marcadores.

Assim, de acordo com o que é especificado em [10], todo o tráfego que entre numa rede *DiffServ* é classificado e condicionado nas fronteiras da rede, onde é atribuído a diferentes agregados e depois é processado internamente por agregado.

O trabalho do IETF focou-se na altura em dois tipos de serviços: o *Assured Service*, também chamado de *Assured Forwarding Service*, e *Premium Service*, ou *Expedited Forwarding Service*.

2.1.2.1 Campo de Serviços Diferenciados

Para que se pudesse diferenciar os pacotes e criar um comportamento diferenciado, o IETF propôs que fosse usado um byte do cabeçalho do protocolo IP. Tanto no IPv4 (campo ToS), como no IPv6 (campo Class) existe um byte que suporta QdS. O campo proposto chama-se *Differentiated Service* (DS) e foi definido de modo a manter compatibilidade com esse byte. O *Differentiated Service Code Point* (DSCP) é assim inserido no campo Class (IPV6) ou no campo *TOS* (IPv4), como se pode ver na Figura 7.

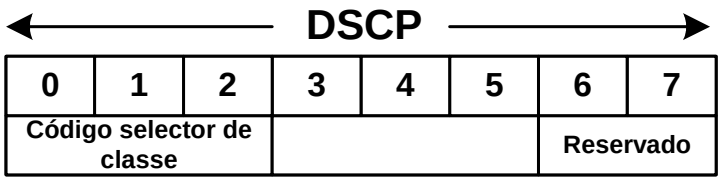

Figura 7 - Valor DSCP no campo Class

O DSCP define qual o serviço que o pacote recebe na rede, bem como o tratamento que deve ter por parte dos *routers*. Vários DSCP podem ser agrupados e receber o mesmo PHB nos nós DS. Os pacotes de um fluxo têm o mesmo valor de DSCP.

2.1.2.2 Serviços Diferenciados

Como serviços diferenciados foram definidos:

- ***Best-Effort Service.*** É o tipo de serviço em que assenta a Internet actualmente. Não tem qualquer diferenciação de QdS. Todos os utilizadores podem tentar dispor dos recursos que a rede tem sem qualquer discriminação. Para este serviço definiu-se o code point 000000 (*default code point*).
- ***Expedited Forwarding Service (EF)***. É também designado como *Premium Service* e é entendido como uma linha alugada virtual. Neste serviço a Largura de Banda não pode ser excedida, apesar de poder ser total ou parcialmente usada. Neste serviço o cliente não deve sentir a influência da possível simultaneidade de utilização da rede por outros utilizadores. O codepoint recomendado é o 101110.
- ***Assured Forwarding Service (AF)***. É normalmente também conhecida como *Assured Service*. Fornece uma Largura de Banda aos utilizadores, apesar de não garantir a sua possibilidade de utilização. Os seus pacotes são marcados com uma prioridade mais alta do que os pacotes de *best-effort* e a rede funciona de modo que, em condições de congestionamento da rede, os fluxos deste serviço sintam sempre uma menor redução de largura de banda do que os utilizadores do serviço *best-effort*. Existem quatro classes, cada uma com três níveis de precedência de descarte. Cada uma é atendida separadamente das outras, tendo a sua própria fila de espera. Os DSCP definidos para as quatro classes de acordo com a precedência de descarte podem ser vistos na Tabela 1.

Precedência de descarte	Classe 1	Classe 2	Classe 3	Classe 4
Baixa	001010	010010	011010	100010
Media	001100	010100	011100	100100
Alta	001110	010110	011110	100110

Tabela 1 - Codepoints de AF

2.1.2.3 Elementos da arquitectura *DiffServ*

Um nó *DiffServ* pode ser representado por uma combinação de quatro componentes cuja relação pode ver-se pelo esquema da Figura 8.

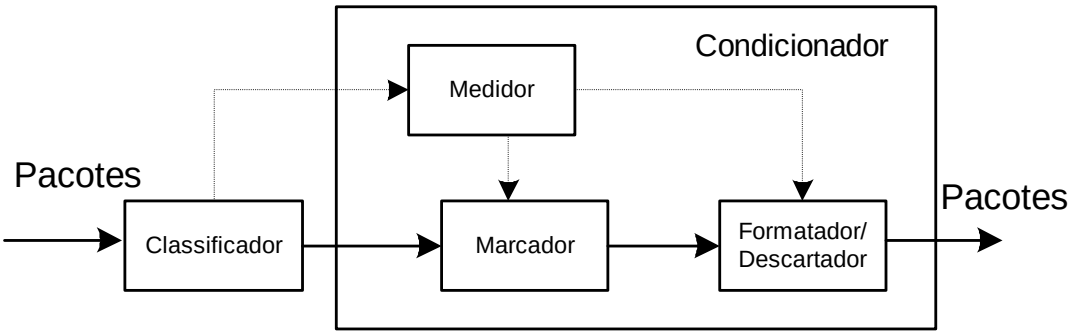

Figura 8 - Classificador e condicionador de tráfego

Quando o pacote chega ao classificador é-lhe atribuída uma classe de acordo com a categoria do serviço a prestar. Depois o classificador envia o pacote para o *condicionador,* que inclui um *medidor*, um *marcador*, um *formatador* e um *descartador*. As linhas contínuas representam o percurso dos pacotes, enquanto as linhas a tracejado representam sinais de controlo enviados para os componentes do *condicionador*.

As características de cada elemento são as seguintes:

- ***Classificador***. Podem existir dois tipos de classificador: classificador de *Behaviour Aggregate* (BA), que escolhe os pacotes só com base nos *codepoints*, e o classificador *Multi-Field*, que escolhe com base em outra informação, que pode ser os endereços de origem e destino, o ID do protocolo, os portos ou outra.
- ***Medidor***. Mede durante algum tempo os parâmetros de um fluxo de pacotes e compara esses dados com o perfil de tráfego especificado no *Traffic Conditioning Agreement*. Depois passa informação aos outros blocos de condicionamento, para que saibam da concordância dos parâmetros de um fluxo com o perfil correspondente.
- ***Marcador***. Adiciona ou altera o valor do campo DS ao pacote, juntando-o aos pacotes de um determinado serviço.
- ***Formatador***. É uma entidade que faz armazenamento de pacotes e que é normalmente controlado pelo medidor. Atrasa um ou mais pacotes de um fluxo, para que este respeite as condições do perfil de tráfego.

- ***Descartador***. Elimina um ou mais pacotes de um fluxo, para que este fique em conformidade com o perfil de tráfego. Este procedimento é conhecido como policiamento e pode ser implementado no Formatador.

2.1.2.4 Service Level Agreements e Traffic Conditioning Specifications

Um *Service Level Agreement* (SLA) é um acordo legal que contém a informação necessária para definir os parâmetros do serviço. Um SLA é definido pelos parceiros (institucionais) envolvidos no acordo, não sendo transmitido aos elementos que gerem a rede.

O *Service Level Specification* (SLS) corresponde à componente técnica do SLA, que pode especificar, entre outros requisitos: a disponibilidade do serviço, descrevendo acções a serem realizadas em situações de descontinuidade do mesmo; os mecanismos de autenticação e criptografia empregados; as restrições de rotas a serem utilizadas no encaminhamento do tráfego agregado; as formas de contabilização e auditoria da QdS fornecida [28]. Mais especificamente, o SLS define que o tráfego de uma determinada classe, respeitando determinadas condições, que entre por um determinado link, vai ser tratado de acordo com conjunto de PHB. Se o destino do tráfego estiver fora do referido domínio, vai ser entregue a outro domínio (de acordo com a tabela de encaminhamento) onde exista um SLS que especifique um conjunto equivalente de PHB's.

O *Traffic Conditioning Specification* (TCS) é a estrutura de rede, correspondente a uma dada SLS, que define os parâmetros do serviço como os PHB, os parâmetros para os condicionadores, para os *policers*, para os marcadores e para os formatadores de tráfego. Define ainda os parâmetros de QdS, como a Largura de Banda (LB), probabilidade de descarte, prioridade, atraso, etc.

A Figura 9 ilustra a relação entre os três conceitos.

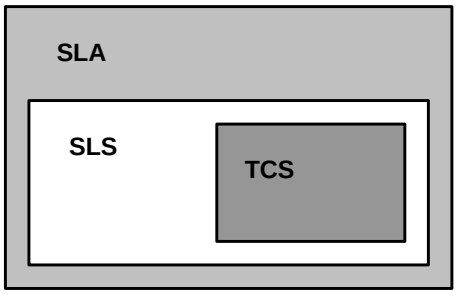

Figura 9 - Relação entre SLA, SLS e TCS

2.1.3 Comutação de Etiquetas Multiprotocolo - MPLS

Em Abril de 1997 o IETF criou um grupo de trabalho com o objectivo de criar uma tecnologia padrão de comutação de etiquetas e agregar algumas tecnologias existentes:

- Cell Switching Router da Toshiba;
- IP Switching da Ipsilon;
- Tag Switching da Cisco;
- Aggregate Route-based Switching da IBM.

A Comutação de Etiquetas Multiprotocolo (*MultiProtocol Label Switching* [11]) foi a designação acordada pelos fabricantes para esta tecnologia. O termo Multiprotocolo advém do facto de toda a arquitectura se poder aplicar a qualquer protocolo da camada de rede. Exemplos de aplicações de MPLS sobre diferentes protocolos da camada de rede são ATM, Frame Relay e IP.

Embora o MPLS não seja estritamente uma tecnologia de QdS, dispõe de mecanismos de diferenciação de QdS.

2.1.3.1 Introdução

Enquanto um pacote de uma rede IP viaja de um router para outro, cada router toma uma decisão de encaminhamento independente de todos os outros. Cada router analisa o cabeçalho do pacote (Figura 10) e corre um algoritmo de encaminhamento que determina qual deve ser o próximo nó para esse datagrama, através da análise das tabelas de encaminhamento que possui.

Os cabeçalhos dos pacotes IP contêm bastante mais informação do que a necessária para determinar qual deve ser o próximo nó, o que por si dificulta o processo de análise dos valores relevantes.

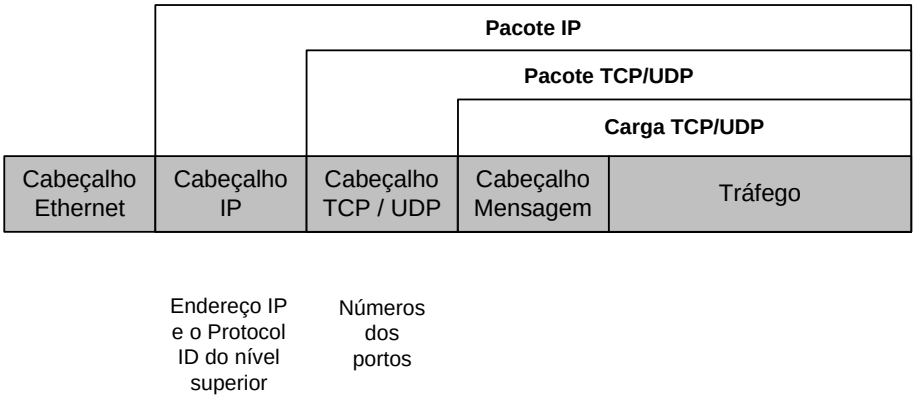

Figura 10 - Informação analisada para determinar o próximo nó numa rede IP

O conjunto de acções executadas com vista a reencaminhar um pacote para o próximo nó podem ser subdivididas em dois grupos: a classificação do pacote tendo em conta, por exemplo, os endereços de destino e origem e, por ventura, informação extra como os portos e protocolo de transporte; e a determinação do próximo nó que melhor reencaminha o pacote de acordo com a classificação determinada.

O paradigma de reencaminhamento do MPLS determina que dentro de uma rede MPLS, uma vez feita a classificação de um determinado pacote, essa acção não seja mais executada, sendo a escolha do próximo nó para onde reencaminhar feita com base na classificação efectuada no nó de entrada. Após a classificação de um pacote, um router MPLS atribui-lhe uma etiqueta e reencaminha-o para o próximo nó. Este router, depois de analisar a etiqueta, usa-a como índice da tabela que determina qual o próximo nó e qual a próxima etiqueta que o pacote deve usar posteriormente. Troca a etiqueta e reencaminha-o para esse nó.

Este paradigma de reencaminhamento tem várias vantagens:

- permite que o reencaminhamento seja feito por equipamento de camada 2 que consegue analisar etiquetas e substituí-las, mas que não seria capaz de analisar cabeçalhos da camada de rede, ou que pelo menos não o faria com uma rapidez adequada;
- permite diferenciar pacotes que entram na rede em diferentes pontos;
- permite que os critérios que definem a classificação de um pacote possam ser muito elaborados sem que se tenha que suportar essa complexidade de análise em todos os nós. Uma vez que o pacote só é classificado na entrada da rede, essa classificação pode incluir análise de informação de camadas de mais alto nível, como informação dos portos;
- pode facilmente definir qual é o trajecto completo de um pacote por questões de segurança ou políticas;
- torna a aplicação de políticas de QdS uma tarefa muito simples.

2.1.3.2 Comutação de etiquetas

Quando um pacote chega a um domínio de *Label switching* é classificado (por exemplo através do endereço de destino ou do porto) por um router que é denominado de *Label Edge Router* (LER). Depois com base na classificação o router de ingresso atribui-lhe uma etiqueta. A partir daí, enquanto o pacote se encontrar dentro do domínio, o encaminhamento é feito apenas com base na informação presente na etiqueta. A etiqueta vai ser retirada ao pacote quando este atingir o router de egresso: o router a que está ligado o endereço destino, ou o router fronteira da rede do domínio *Label Switching*. Designa-se por *Label Switching Path* o caminho a seguir pelos pacotes com uma mesma etiqueta, o que é equivalente a um circuito virtual de uma rede ATM. O processo é ilustrado na Figura 11.

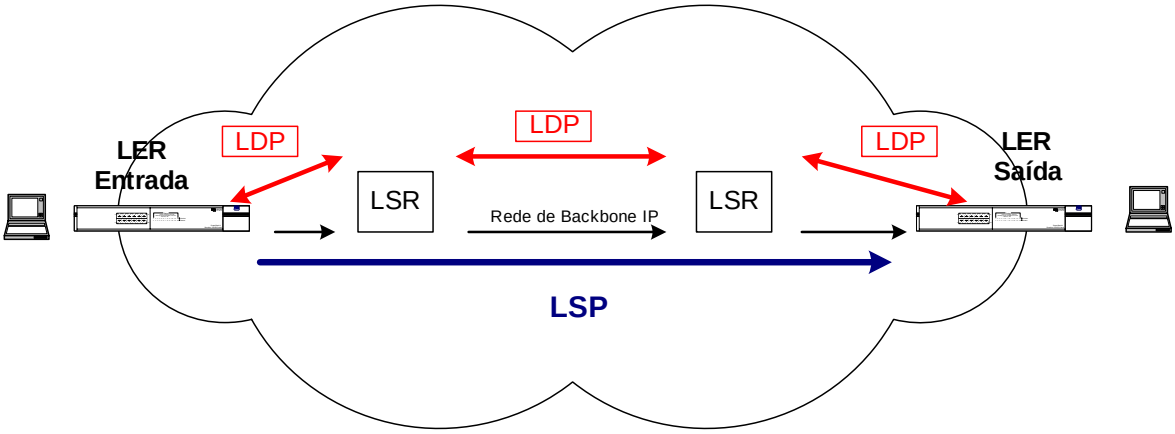

Figura 11 - Rede MPLS

2.1.3.3 Forwarding Equivalence Class

Uma *Forwarding Equivalence Class* (FEC) designa um grupo de pacotes com características comuns. Estas características podem ser o endereço destino, o endereço origem, ou os portos. Todos os pacotes pertencentes à mesma FEC são tratados da mesma maneira. A associação de um pacote a uma FEC faz-se através da introdução de uma etiqueta. A etiqueta introduzida no pacote indica qual a FEC a que este pertence e é usada pelos nós intermédios para encaminhar o pacote. Exemplos de tráfego que podem ser agregados com uma etiqueta [12]:

- ***Port Quadruples***: mesma subnet de origem, mesma subnet de destino, mesmo TTL, protocolo IP e portos TCP/UDP destino;
- ***Port Quadruples com TOS***: equivalente ao anterior, mas com a restrição dos pacotes terem o mesmo TOS;
- ***Pares de terminais***: como mesmo endereço destino e origem;
- ***Pares de Redes***: Mesmas subnet de origem e destino;
- ***Redes de destino***: mesmas redes de destino;
- ***Router de Egresso***: mesmo router de entrada;
- ***Next Hop AS***: Sistema autónomo do próximo nó;
- ***AS destino***: Sistema autónomo do destino.

2.1.3.4 Label Distribution Protocol

O protocolo de distribuição de etiquetas - *Label Distribution Protocol* (LDP) - encarrega-se de distribuir a informação entre os Label Switching Routers (LSR) e os LER. Protocolos como o RSVP e BGP contêm também suporte para poderem participar na distribuição de etiquetas.

O LDP pode ser usado entre LSR adjacentes ou não adjacentes. Dois LSR comunicam entre si através de sessões de LDP, por isso se designam de LDP peers (LSP).

Existem quatro categorias de mensagens LDP:

- **Discovery**: são mensagens usadas para anunciar e manter a presença de um LSR na rede;
- **Session**: são usadas para estabelecer, manter e terminar sessões de LDP entre LDP peers;
- **Advertisement**: são usadas para criar, mudar e apagar o mapeamento de etiquetas de FEC's;
- **Notification**: são usadas para troca de informação sobre o *status*, diagnósticos e informação de erros entre LDP peers.

Sempre que os LSR necessitam de anunciar ou trocar etiquetas, estabelecem entre si sessões LDP. Cada anúncio, cada troca de etiquetas, requer o estabelecimento de uma sessão específica. As sessões estabelecem-se usando TCP e sempre que são estabelecidas entre LSR não adjacentes são estabelecidos túneis.

O LDP estabelece um conjunto de regras para o mapeamento de um pacote a um LSP:

- se existir exactamente um LSP que tenha uma FEC com o endereço de terminal que seja o mesmo do pacote, então o pacote é mapeado para esse LSP;
- se existirem vários LSP's em que cada um contenha uma FEC com o endereço destino que seja o mesmo do pacote, então o pacote é mapeado para um destes LSP's. A escolha do LSP não é definida pelo LDP;
- se não existir um LSP definido exactamente até ao destino, o pacote deverá ser mapeado para aquele que contenha o prefixo de endereço de destino mais semelhante ao seu;
- se for conhecido que o pacote deverá atravessar um dado LSR de egresso e se existir um LSP com uma FEC com o endereço desse LSR, então o pacote deverá ser mapeado para esse LSP.

2.1.3.5 O cabeçalho de MPLS

O cabeçalho MPLS ilustrado na Figura 12 é criado pelo LSR de Ingresso e é constituído pelos seguintes campos:

- **Etiqueta**: contém o valor da Etiqueta que é analisada pelo percurso efectuado pelo pacote e que ocupa 20 bits;

- **Exp**: campo experimental com 3 bits, igual ao usado na arquitectura *DiffServ* com a designação de DSCP, que indica a classe do pacote encapsulado. Este campo pode ser usado num domínio MPLS para suportar *DiffServ*;
- **S bit**: bit que identifica a presença de múltiplas etiquetas;
- **TTL**: substitui o TTL do datagrama IP que não sendo analisado ao longo do caminho pelos LSR não pode ser decrementado por estes, e assim usam este TTL para esse fim.

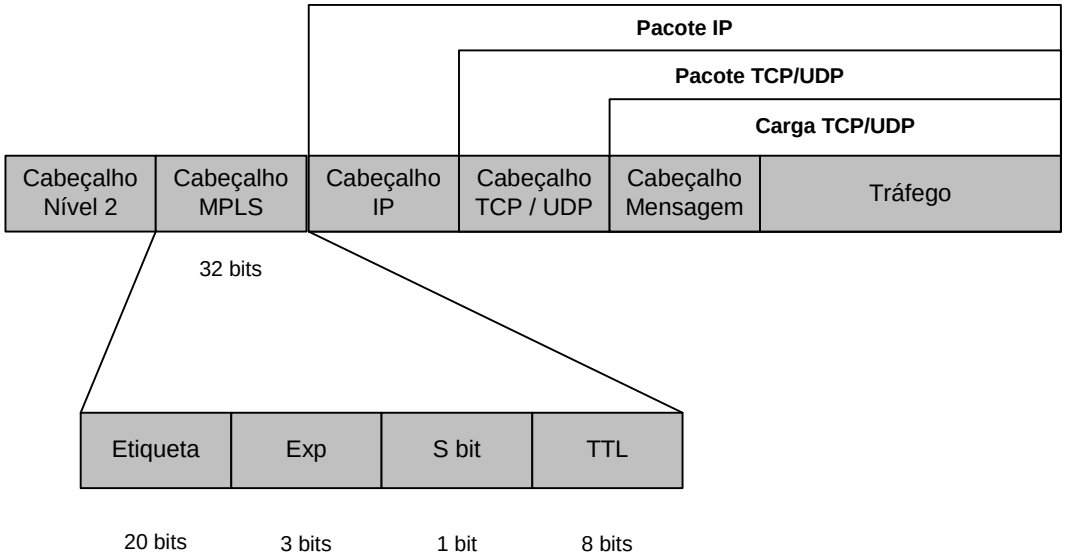

Figura 12 - Cabeçalho MPLS

Este cabeçalho é colocado entre os cabeçalhos de nível 2 e de nível 3 (de IP).

2.1.4 Integração das arquitecturas Integrated Services e Differentiated Services

A arquitectura *IntServ* (capítulo 2.1.1) tem sérios problemas de escalabilidade, apesar de poder fornecer garantias de QdS entre extremos. Em relação à arquitectura *DiffServ* passa-se precisamente o oposto; é uma arquitectura escalável, mas que precisa de mecanismos complexos para poder fornecer garantias de QdS fim-a-fim. A arquitectura *IntServ* é especialmente vocacionada para redes de acesso pelos seus mecanismos de controlo de admissão, enquanto que a arquitectura *DiffServ* é vocacionada para redes de core. A integração de ambas as arquitecturas permite aproveitar as melhores características de cada uma, sem que se sofram as consequências dos seus defeitos.

Apesar das vantagens criadas pela integração das arquitecturas, criam-se alguns desafios com que é preciso lidar:

- é necessário adaptar as classes de serviço de cada uma das arquitecturas de modo a que o tráfego vindo de uma delas pertencente a uma classe de serviço seja transportado pela rede da nova arquitectura com uma classe que seja equivalente à anterior;
- a sinalização RSVP que indica os recursos pretendidos por cada fluxo não é interpretada pela rede *DiffServ*. Do mesmo modo, o tráfego da rede *DiffServ* não possui sinalização RSVP que tem que ser criada no router de fronteira com a rede *IntServ*, ou então ser transportado pela rede *DiffServ* até que seja interpretado quando der entrada na rede *IntServ*.

A RFC 2998 [13] propõe uma plataforma de suporte de *IntServ* em redes *DiffServ*. As secções seguintes discutem os conceitos aqui apresentados.

2.1.4.1 Mapeamento de serviços

Um pedido de serviço *IntServ* define um conjunto quantitativo de parâmetros descritos pelo *flowspec*. Cada nó numa região *IntServ* interpreta o pedido por forma a decidir se tem recursos disponíveis para poder aceitar esse pedido. Numa rede *DiffServ* não existe qualquer sinalização para requisitar os recursos, é o valor do campo DSCP define qual o serviço pretendido. Existe então necessidade de se mapear os serviços pedidos na rede *IntServ* nos serviços disponíveis na rede *DiffServ*. A plataforma de integração proposta na RFC 2998 propõe então:

- que se escolha o PHB, ou conjunto de PHB's existentes na rede *DiffServ* que irá ser usado pelo serviço da rede *IntServ*;
- policiamento adequado nas fronteiras da rede *DiffServ*, incluindo remarcação de pacotes;
- exportação de parâmetros *IntServ* da rede *DiffServ* através da actualização da informação presente no ADSPEC;
- que se tenham em conta os recursos disponíveis na rede *DiffServ* no controlo de admissão pela rede *IntServ*.

É assim necessário que a rede seja coerente no mapeamento dos serviços e que todos os elementos da rede conheçam a equivalência definida entre os serviços. São propostas várias soluções para o mapeamento de serviços:

- ***mapeamento estático***: é usado um mapeamento cuja relação entre os serviços das diferentes arquitecturas é estática e conhecido por todos os elementos da rede;
- ***mapeamento negociado pelos routers DiffServ***: os routers de entrada na rede *DiffServ* fazem marcação dos pacotes provindos da rede *IntServ*, marcação essa que pode ser alterada pelos routers de fronteira. Nesta solução é proposto o uso de RSVP para a negociação entre os routers *DiffServ*.

2.1.4.2 Gestão de recursos nas regiões *DiffServ*

Existem várias opções para a gestão de recursos na região da rede *DiffServ*:

- aprovisionamento estático de recursos;
- aprovisionamento dinâmico usando RSVP;
- aprovisionamento dinâmico usando um Bandwidth Broker.

Cada uma destas opções apresenta consequências diferentes.

2.1.4.2.1 Aprovisionamento Estático

Num cenário deste género os elementos de rede na região *DiffServ* têm capacidades de comunicar através de RSVP. Os recursos na rede *DiffServ* são aprovisionados estaticamente sendo negociado pelos clientes da rede *DiffServ* e pelo operador da rede um determinado contrato de utilização de recursos - SLS - que estaticamente define quantidades de recursos da rede *DiffServ* reservadas à utilização dos fluxos provindos da rede *IntServ*. A capacidade de transmissão negociada pode ser simplesmente uma quantidade de largura de banda, ou um conjunto mais complexo de parâmetros como taxas de pico, alturas do dia, dia da semana. Neste caso, cada router de fronteira actua como um nó *IntServ* de um lado (onde liga à rede cliente) e como um nó *DiffServ* que liga à rede *DiffServ*. A metade *IntServ* do router fronteira é capaz de identificar e processar o tráfego fluxo a fluxo. A metade *DiffServ* pode ser considerada como um conjunto de interfaces de transmissão, uma por cada serviço *DiffServ*. O router contém uma tabela definindo a capacidade de transmissão definida por cada serviço *DiffServ* negociado, que é usada para fazer controlo de admissão dos fluxos que entram na região *DiffServ*. O cenário encontra-se ilustrado na Figura 13.

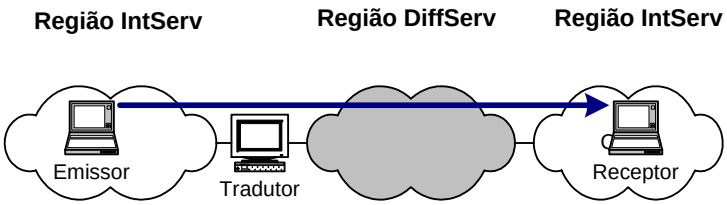

Figura 13 - Esquema de implementação de aprovisionamento estático na rede *DiffServ*

Encontra-se em [14] uma implementação desta solução. Propõe um tradutor de pedidos *IntServ*/RSVP para *DiffServ* que é localizado nos routers de fronteira da região *DiffServ*.

À entrada da região *DiffServ* os pedidos RSVP da rede *IntServ* de partida são interpretados e a classe de serviço da rede *DiffServ* que mais se adequar ao serviço pedido pela rede *IntServ* é procurada. O elemento de fronteira conduz esses pedidos para a rede *DiffServ* como tráfego *best-effort* que, apesar de não ser interpretado é conduzido até à rede *IntServ* de chegada para que as reservas necessárias para o fluxo aí sejam feitas. Todo o tráfego pertencente a cada fluxo é inserido na rede *DiffServ* com a marcação do código DSCP calculado para esse fluxo.

À chegada à rede *IntServ* destino não é feita qualquer tradução, uma vez que os pedidos RSVP são conduzidos até aí e a marcação DSCP também não é removida, uma vez que é ignorada na rede *IntServ*. Assim o pacote encontrará uma reserva feita nessa região e será servido de acordo com essa reserva.

2.1.4.2.2 Região *DiffServ* com suporte RSVP

No caso de os routers pertencentes à região *DiffServ* da rede possuírem suporte RSVP, os routers de fronteira são routers RSVP. Podem ainda existir elementos do interior da região *DiffServ* que o suportem. Apesar de todos estes routers participarem na sinalização RSVP os routers do interior da rede *DiffServ* classificam e escalam o tráfego de acordo com o código DSCP e não com os critérios de classificação por fluxos normais dos routers RSVP/*IntServ*. Esta aproximação explora as vantagens da sinalização RSVP, enquanto mantém grande parte da escalabilidade do *DiffServ*.

Na alternativa apresentada na secção 2.1.4.2.1 não existia sinalização entre elementos da região *DiffServ* e de fora dela. O router de entrada da região *DiffServ* actuava como agente de controlo de admissão para a rede *DiffServ*. A negociação da SLS era estática, o que dificulta a gestão de recursos na rede *DiffServ* pelo facto de os critérios de admissão serem os parâmetros negociados na SLS e não a disponibilidade de recursos existentes na rede. Quando a rede *DiffServ* tem capacidades RSVP o agente de controlo de admissão é parte da rede *DiffServ* e, por esse facto,

as mudanças na quantidade de recursos disponíveis na rede *DiffServ* podem ser comunicadas para fora da rede *DiffServ* através do uso de RSVP. Incluindo routers *DiffServ* com capacidades RSVP, é possível aumentar a eficiência e ao mesmo tempo aumentar as garantias de que os recursos atribuídos pelos mecanismos de controlo de admissão estão realmente disponíveis. Esta melhoria nas garantias de QdS acontece porque os mecanismos de controlo de admissão podem analisar a quantidade de recursos disponíveis ao longo de todo o trajecto que o fluxo a ser admitido vai percorrer. O RFC 2998 descreve esta vantagem de sinalização do RSVP como "*controlo de admissão ciente da topologia*".

Outra vantagem do suporte RSVP nos elementos de rede *DiffServ* é o facto de se poder efectuar alterações no aprovisionamento da rede *DiffServ*, por exemplo, através da alocação de mais ou menos largura de banda para uma determinada classe de serviço em função dos pedidos de recursos provenientes de fora da rede *DiffServ*.

2.1.4.2.2.1 Agregação de RSVP

O RSVP pode ser usado de modo a fazer reserva de recursos para agregados de fluxos entre extremos de uma rede. Os routers de fronteira podem interagir com os routers do interior da rede *DiffServ*. Os valores das reservas iniciais podem ser definidos pelos router de fronteira de acordo com o tipo de tráfego que anteriormente circulou na rede, podendo ser feitas alterações nos valores das reservas de acordo com as variações do tráfego que lhe seja oferecido na rede de acesso com o decorrer do tempo.

Com esta aproximação, o controlo de admissão dos pedidos dos fluxos RSVP oferecidos aos nós exteriores à região RSVP seria feito de acordo com os recursos disponíveis de cada classe de serviço dentro da região RSVP. O tamanho das reservas para os agregados poderia ou não ser ajustado para se adaptar às mudanças das reservas dos fluxos e uma taxa relativamente lenta. As vantagens deste tipo de solução são a flexibilidade, o controlo de admissão ciente do estado de sobrecarga da rede *DiffServ* sem necessitar da quantidade de sinalização e processamento RSVP que existe na arquitectura *IntServ*.

A gestão de recursos numa região *DiffServ* usando RSVP agregado é mais conveniente dentro de um só domínio administrativo, uma vez que cada domínio administrativo tende a escolher mecanismos diferentes de gestão de recursos, e cuidados adicionais teriam que ser mantidos em todos os domínios.

2.1.4.2.2.2 RSVP por cada fluxo

Segundo esta aproximação os routers de uma região *DiffServ* responderiam à sinalização RSVP originada pelos nós *IntServ* de fora da região *IntServ*, por cada fluxo. Esta solução apresentaria as vantagens da solução anterior com muito dinamismo no aprovisionamento de recursos da rede *DiffServ* sem a necessidade do suporte de agregação do RSVP. Os recursos seriam usados ainda mais eficientemente como resultado da admissão de controlo fluxo a fluxo. Contudo, a quantidade de processamento e de sinalização RSVP pode ser significativamente maior do que no caso de uso de agregação do RSVP.

2.1.4.2.3 Aprovisionamento Dinâmico, sem suporte de RSVP na região *DiffServ*

Os routers de fronteira podem não usar nenhuma forma de sinalização RSVP para a região *DiffServ* e, em vez disso, usar outro tipo de protocolo que interaja com um oráculo [13], como ilustrado na Figura 14. O oráculo seria um agente que teria suficiente conhecimento da disponibilidade dos recursos da rede e da topologia da mesma que poderia tomar decisões de controlo de admissão. Neste caso o oráculo, que actuaria como um BB, poderia substituir o conjunto de routers RSVP dos exemplos anteriores na tomada de decisões.

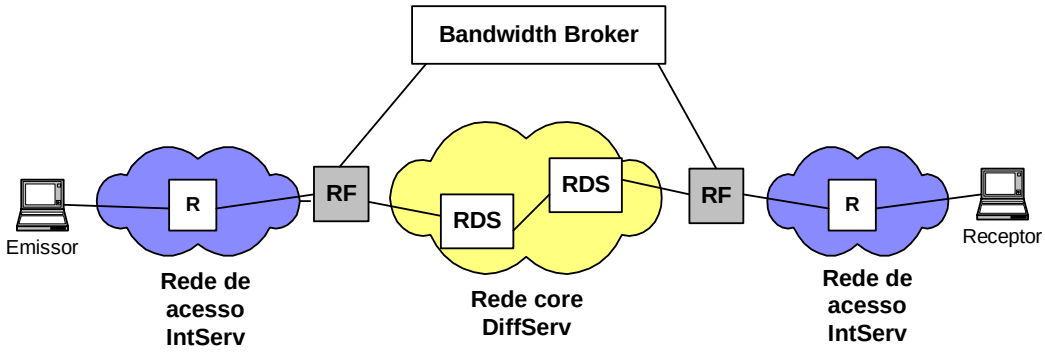

Figura 14 - Utilização de BB para controlo de admissão na rede *DiffServ*

Existem várias implementações deste cenário[20], [16]. Fazem uso de um BB para fazer controlo de admissão na rede *DiffServ*, por forma a poderem dar garantias de QdS entre extremos. Possuem ambos uma rede de acesso *IntServ* onde os routers de fronteira fazem a tradução dos serviços da rede *IntServ* para as Classes de serviço (CdS) da rede *DiffServ*. Em [16] faz-se uso do protocolo COPS para implementar a comunicação entre os routers de fronteira e o BB, enquanto que em [20] foi implementado um protocolo proprietário. Nos dois casos os BB podem criar uma reserva na rede *IntServ* para alocação de um novo fluxo.

Ambas as soluções anteriormente citadas apresentam problemas de escalabilidade:

- é previsto um único BB para fazer o controlo de admissão em toda a rede *DiffServ*;
- a sinalização RSVP, apesar de não ser processada dentro da rede *DiffServ*, é por ela transportada, uma vez que volta a ser necessária na rede *IntServ* destino para que a reserva para o recurso seja feita assim que este lá chegue.

Em [21], no âmbito deste trabalho, é proposta uma solução que resolve estes problemas. Os routers de fronteira possuem capacidades de marcação de pacotes, bem como de criação e interpretação de sinalização RSVP. Podem eliminar a sinalização RSVP à entrada da rede *DiffServ* e voltar a adicioná-la no elemento de fronteira com uma nova rede *IntServ*, evitando assim uma grande quantidade de tráfego inútil.

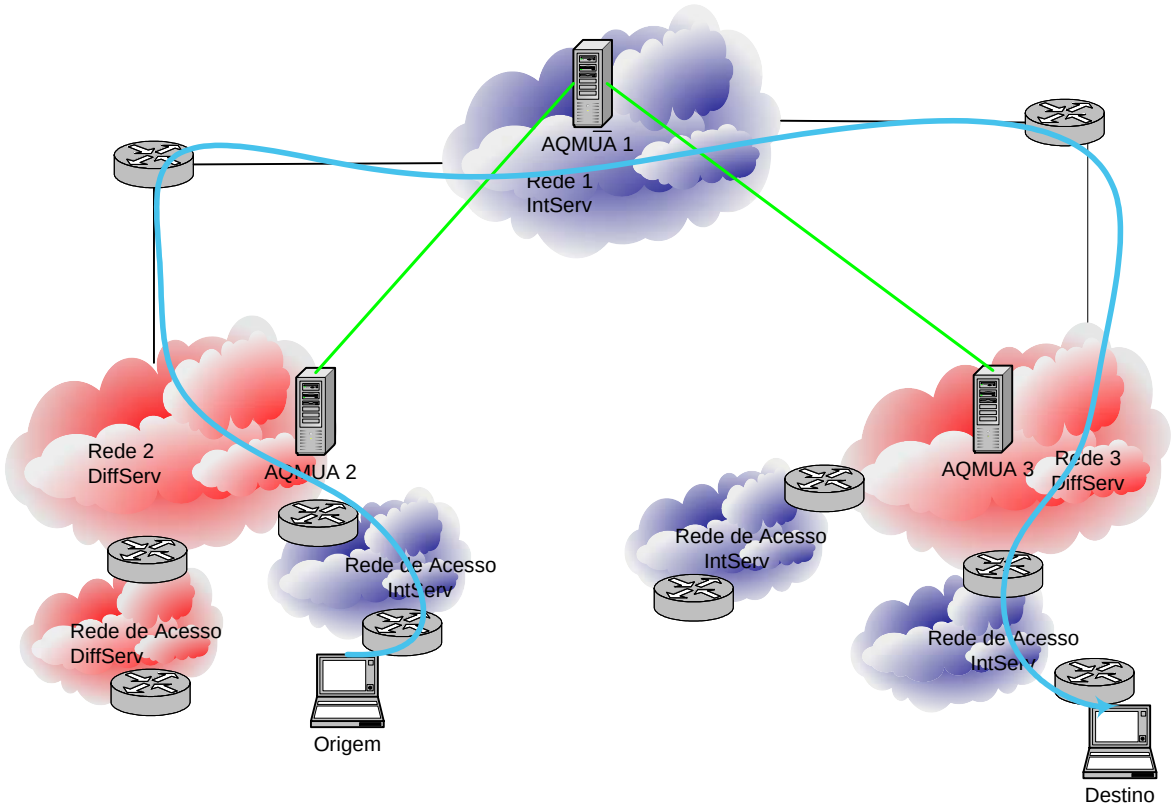

Figura 15 - Exemplo de adaptação do tráfego entre redes de diferentes arquitecturas

A proposta prevê que os BB possam estabelecer uma rede pela qual trocam informação acerca da topologia e arquitectura da rede que gerem. Esta solução permite conduzir tráfego por várias redes com diferentes arquitecturas de QdS de uma forma mais flexível, não sendo necessário que coexistam nas redes *DiffServ* marcação de pacotes e sinalização RSVP. A arquitectura proposta permite ainda o controlo de admissão nas redes *DiffServ*, permitindo um controlo integrado de recursos nas redes de diferentes arquitecturas e assim permitindo fornecer garantias de QdS entre extremos.

A utilização de um router de acesso com suporte de QdS de várias arquitecturas permite a criação de uma rede de acesso com uma arquitectura heterogénea, onde coexistam máquinas que façam marcação de pacotes com máquinas que façam pedidos RSVP ao router de acesso. A integração das duas arquitecturas numa só rede de acesso permite a reutilização de máquinas, cujo software de suporte de QdS não estava adaptado à arquitectura da rede onde ia ser ligada.

O router proposto permite ainda fornecer QdS para aplicações que não estejam preparadas para a requisitar, podendo fazer a marcação de pacotes ou criação de sinalização RSVP de acordo com um conjunto de políticas fornecidas pelo BB. Esta característica permite reutilizar software desenvolvido sem ter em conta requisição de QdS sem que este tenha que ser rescrito.

As capacidades de marcação de marcação e criação de pacotes aliadas à utilização de uma rede de BB poderiam ainda permitir criar suporte para multicast. O BB funcionaria como um gestor de subscrições para esses serviços, podendo ter grande aplicabilidade em redes UMTS ou WiFI onde os recursos rádio necessitam de ser utilizados de uma forma eficiente devido à sua escassez.

3. CONTROLO DO QDS

3.1 CONTROLO DE ADMISSÃO

Numa rede com QdS, para que se possam dar garantias que os fluxos já aceites irão ser continuamente servidos de acordo com as garantias de QdS correspondentes ao seu serviço, é preciso que haja um mecanismo analisador dos recursos disponíveis na rede e da capacidade para servir um novo fluxo, sem que se degrade a qualidade do serviço prestado aos fluxos já aceites. Este mecanismo chama-se controlo de admissão. Os conceitos de controlo de admissão não são exclusivos de um modelo de QdS em particular; são usados quer na arquitectura *IntServ*, quer na arquitectura *DiffServ*.

Sempre que chega um novo fluxo a uma rede com mecanismos de controlo de admissão são analisados os recursos (tipicamente a largura de banda) disponíveis na rede. Se estes recursos forem suficientes para servir o novo fluxo a ser admitido sem degradar os fluxos que entretanto já estão a ser servidos, o novo fluxo é admitido. Caso contrário é negado. Podem ocorrer situações em que a aceitação de um novo fluxo obriga à interrupção ou degradação de um ou mais fluxos anteriores por questões de prioridade.

Uma das características mais importantes do controlo de admissão é a possibilidade de operar de acordo com uma gestão baseada em políticas. Esta gestão pode incluir a identificação dos utilizadores e de aplicações ou simplesmente analisar somente como, quando e onde o tráfego entra na rede. A RFC 2753 [26] descreve uma arquitectura de controlo baseado em políticas para controlo de admissão.

O controlo de admissão pode ser distribuído de uma forma semelhante ao que acontece com a arquitectura *IntServ* (descrito na secção 2.1.1), ou então pode ser centralizado fazendo uso de uma entidade central que toma as decisões de controlo de admissão e que geralmente faz a gestão de todos os recursos da rede [23]. A secção 3.4 faz uma análise das diferentes arquitecturas conceptuais de controlo da rede.

3.2 CONTROLO DE TRÁFEGO

3.2.1 Escalonamento

O escalonamento levado a cabo pelo escalonador consiste na implementação de um algoritmo que decide a ordem pela qual, num servidor de pacotes (por exemplo, na ligação de saída de um router), são servidos os pacotes pertencentes a diferentes fluxos.

São a seguir brevemente descritos alguns dos algoritmos de escalonamento existentes: *First-In-First-Out*, Prioridade estrita, *Round-Robin*, *Weighted Round-Robin*, o *Generalized Processor Sharing* e o *Weighted Fair Queuing*. Relativamente ao tipo de ordenação utilizado podem ser classificados:

- Por ordem de chegada: FIFO;
- De uma forma rígida (prioridade estrita);
- De uma forma rotativa (*Round-Robin, Weighted Round-Robin e Deficit Round-Robin*);
- Por distribuição ponderada da largura de banda (*Weighted Fair Queuing e Self Clock Fair Queuing*).

3.2.1.1 First - In - First - Out (FIFO)

No algoritmo FIFO os pacotes são simplesmente servidos por ordem de chegada. Este sistema é fácil de implementar, não requerendo qualquer processamento de ordenação. Por outro lado, ao tratar todo o tráfego de igual modo não permite oferecer diferenciação de qualidade de serviço diferenciada. Além disso, o envio seguindo a ordem de chegada leva a que fluxos que gerem n vezes mais tráfego recebam n vezes mais serviço.

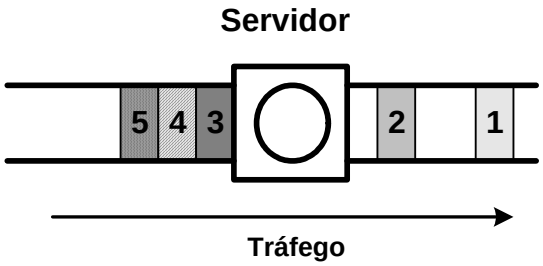

Figura 16 - Ilustração do algoritmo FIFO

3.2.1.2 Prioridade Estrita

O sistema de prioridade estrita assegura que o tráfego classificado como de maior prioridade seja sempre servido em detrimento do classificado como de menor prioridade. Este sistema não requer qualquer processamento de ordenação; no entanto, requer classificação de acordo com a prioridade.

Ao contrário do FIFO, o sistema de prioridade estrita permite diferenciação da qualidade de serviço. No entanto, a diferenciação entre o serviço dado às diferentes prioridades pode ser exagerada: um fluxo de tráfego de maior prioridade, ao enviar uma grande quantidade de tráfego, pode impedir que os fluxos de menor prioridade recebam qualquer serviço.

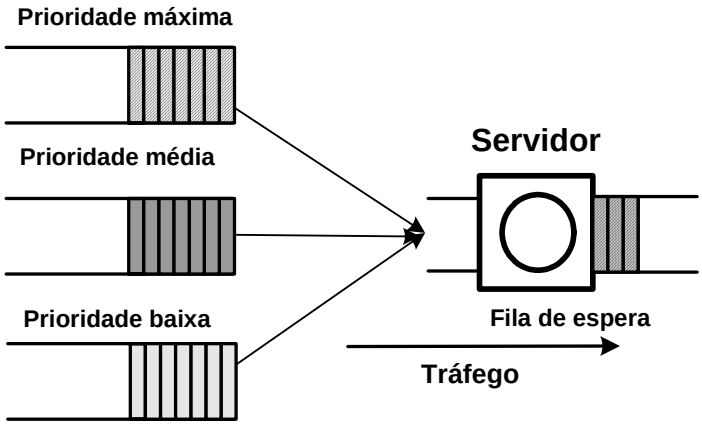

Figura 17 - Ilustração do algoritmo de prioridade estrita

3.2.1.3 Round - Robin (RR)

O algoritmo *Round-Robin* selecciona um pacote de cada fila de forma rotativa. Num sistema com um número de filas de espera (k), o algoritmo transmite sequencialmente pacotes das filas 1 a k, completando um ciclo, após o que repete o procedimento anterior, isto é, volta a transmitir sequencialmente pacotes das filas 1 a k, iniciando novo ciclo, e assim sucessivamente.

Este sistema não permite, por si só, diferenciação de serviço, uma vez que trata todas as filas de espera de igual forma. No entanto, ao separar os diversos fluxos de tráfego em filas de espera independentes, permite um certo grau de protecção. Servindo as filas de forma sequencial, evita que um único fluxo possa usurpar os recursos da ligação, ao contrário do que acontece com o FIFO, onde fluxos que gerem n vezes mais tráfego recebem n vezes mais serviço. Apesar deste grau de protecção, o acto de servir apenas um pacote de cada fila beneficia os fluxos com pacotes

maiores, pelo que o algoritmo *Round-Robin* simples apenas se adequa a redes que utilizem pacotes de tamanho fixo.

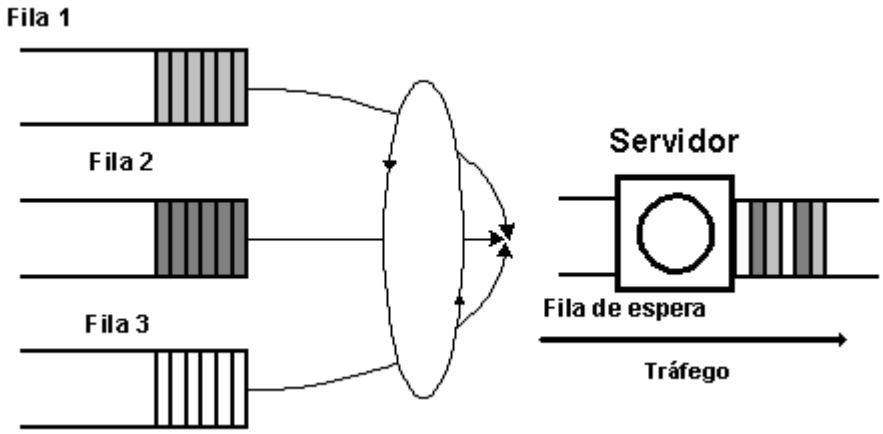

Figura 18 - Ilustração do algoritmo *Round-Robin*.

3.2.1.4 Weighted Round-Robin (WRR)

No algoritmo *Weighted Round-Robin* é atribuído um peso φ_i a cada fila de espera. Em cada ciclo, o algoritmo serve uma quantidade de tráfego em octetos de cada fila proporcional ao peso respectivo. O sistema calcula o número mínimo de pacotes de cada fila a servir em cada ciclo, por forma a que o produto deste número pelo comprimento médio dos pacotes esteja na proporção do peso das filas.

Este algoritmo tem duas desvantagens: (i) é necessário conhecer à partida uma estimativa de qual o tamanho médio dos pacotes e (ii) o servidor pode ficar demasiado tempo a servir o mesmo fluxo de pacotes.

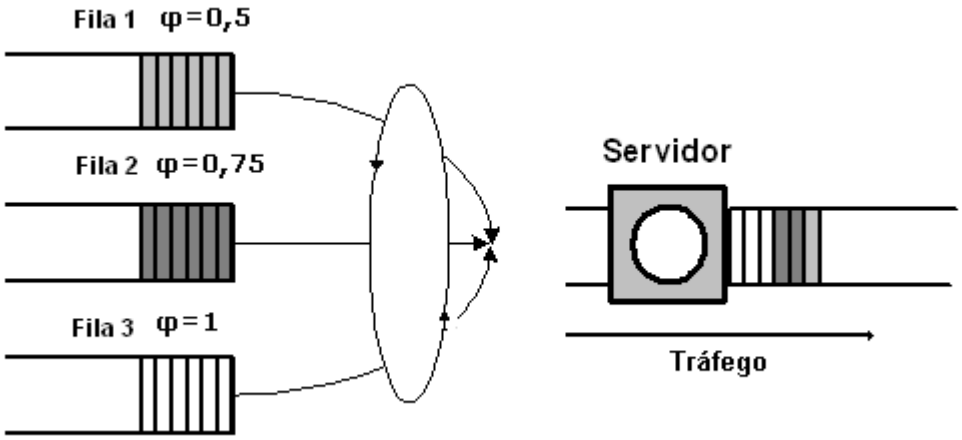

Figura 19 - Ilustração do algoritmo *Weighted Round Robin*.

3.2.1.5 Deficit-Round-Robin (DRR)

O algoritmo *Deficit-Round-Robin* actua distribuindo uniformemente a largura de banda disponível através de um processo independente do comprimento dos pacotes. O DDR procura, em cada ciclo, servir uma quantidade de tráfego em octetos pré-definida que se designa por limiar.

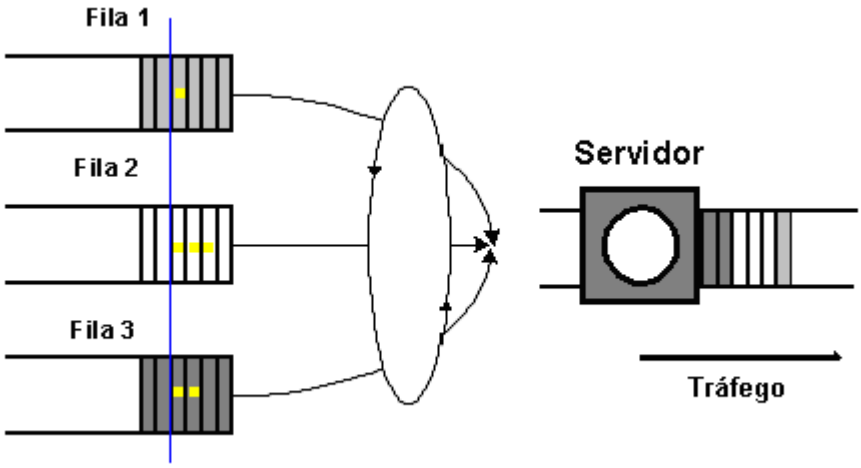

Figura 20 - Ilustração do funcionamento do mecanismo *Deficit-Round-Robin*. Nesta figura, os pacotes assinalados com um ponto são os escolhidos para serem servidos, por se encontrarem a jusante do limiar.

O algoritmo DDR associa a cada fila i um crédito c_i, cujo valor inicial é zero, e um limiar L_i. O limiar é uma constante característica da fila; o crédito pode variar de ciclo para ciclo e só pode tomar valores não negativos.

Em cada visita à fila i o crédito disponível é incrementado de um valor igual ao limiar, ou seja, $c_i = c_i + L_i$. Nessa visita, o sistema pode servir um ou mais pacotes até esgotar o crédito disponível.

Existindo um outro pacote na mesma fila, este pacote pode também ser servido se o seu comprimento for menor ou igual ao crédito disponível; caso contrário, o algoritmo passa à próxima fila.

Obviamente, um pacote só poderá ser servido num dado ciclo se houver crédito suficiente para servir todos os seus octetos. Quando uma fila fica vazia o crédito respectivo é colocado a zero; caso contrário, uma fila poderia estar a acumular créditos indefinidamente conduzindo eventualmente a condições de injustiça.

A distribuição de largura de banda de forma não equitativa pode ser conseguida atribuindo limiares diferentes a cada fila de espera. De notar que, preferencialmente, não deve verificar-se

uma situação em que o valor de todos os limiares seja inferior ao comprimento máximo dos pacotes. Neste caso, surgiriam situações em que não seria servido qualquer pacote num ciclo, obrigando a uma actualização desnecessária dos valores dos créditos e aumentando a complexidade associada aos cálculos.

Este sistema possui, relativamente ao sistema WRR, a vantagem de não ser necessário conhecer à partida o comprimento médio dos pacotes. No entanto, por ser um sistema rotativo, não consegue evitar que o tráfego de cada fluxo seja acumulado e servido todo de uma vez (até ao limiar), o que aumenta as variações do atraso. Este problema acentua-se em fluxos que utilizem uma elevada percentagem da largura de banda (limiares elevados) e que tenham pacotes pequenos.

3.2.1.6 Generalized Processor Sharing (GPS)

O *Generalized Processor Sharing* (GPS) é um algoritmo ideal, baseado num modelo de fluídos em que o tráfego é considerado infinitamente divisível, podendo um mesmo servidor ser utilizado em simultâneo por múltiplos fluxos de pacotes. Por exemplo, num dado instante 50% da largura de banda pode estar a ser utilizada por um fluxo, 30% por outro e 20% por outro.

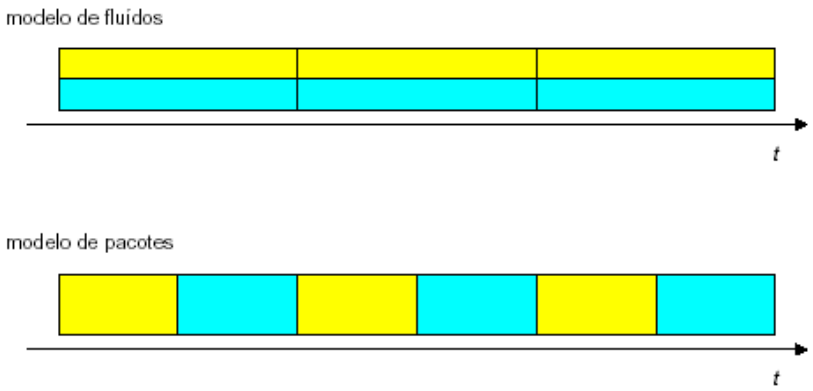

Figura 21 - Modelo de fluídos vs. modelo de pacotes.

O procedimento deste modelo à chegada de um pacote é muito simples. Assim que uma fila de espera começa a receber tráfego, este começa imediatamente a ser servido, em simultâneo com o restante tráfego, a uma taxa proporcional ao seu peso. É um modelo impossível de implementar, mas constitui uma boa base teórica para o desenvolvimento de outros algoritmos, como é o caso do *Weighted Fair Queuing* (WFC), que se descreve a seguir.

3.2.1.6.1 Weighted Fair Queuing (WFQ)

O sistema *Weighted Fair Queuing*, também designado por *Packet Generalized Processor Sharing* (PGPS), é uma aproximação ao sistema ideal GPS. O algoritmo calcula a ordem em que cada um dos pacotes deveriam terminar o serviço. Usa então essa ordem para escolher o pacote que deve ser servido primeiro.

O cálculo necessário a determinar a ordem de entrada em serviço de cada pacote torna este algoritmo muito complexo e muito exigente em termos computacionais. Tem ainda como desvantagens o facto de poder permitir que um pacote seja servido muito mais cedo do que aconteceria no caso de o escalonador ser o GPS, podendo ainda ser muito mais injusto do que seria o GPS.

3.2.1.6.2 Self Clock Fair Queuing (SCFQ)

O algoritmo *Self Clock Fair Queuing* (SCFQ) consiste numa simplificação do algoritmo WFQ, de modo a evitar a grande necessidade de processamento.

A entrada em serviço faz-se, como no WFQ, em função da previsão do instante de terminar o serviço. A simplificação consiste no facto de o algoritmo SCFQ não manter a variável que representa o serviço concluído, tradicionalmente designada por *Round Number*. Isto, que implicava um esforço computacional elevado, usando no cálculo do tempo previsto de saída de serviço de um pacote o tempo previsto do pacote em serviço nessa altura no sistema. Adicionalmente, este algoritmo também não entra em conta com a capacidade da linha, uma vez que a ordem de entrada em serviço de um pacote é independente da linha em causa, só dependendo do seu tamanho, do peso da sua fila e da quantidade de tráfego existente nas outras filas.

Apesar do SCFQ ser fácil de implementar, este algoritmo pode não ser justo para pequenos intervalos de tempo, ao não respeitar as proporções de largura de banda definidas nos pesos. Existem situações em que determinados pacotes são privilegiados perante outros que tem melhores relações tamanho/peso da fila, pelo facto de terem chegado primeiro e por a previsão de tempo de finalização de serviço ser feita quando os pacotes chegam às filas [40].

3.2.2 Gestão de Filas

A gestão de filas é um aspecto também importante, sendo o *Random Early Detection* o algoritmo mais usado.

3.2.2.1 Random Early Detect (RED)

O RED descarta aleatoriamente pacotes de cada fluxo antes que este chegue ao seu limite. O facto de os pacotes serem aleatoriamente descartados antes de que uma determinada fila atinja o seu limite, faz com que num link de backbone congestionado o tráfego abrande de uma forma muito mais suave e evita fenómenos de sincronização de fontes. Ajuda ainda a que o tráfego TCP encontre um ritmo de transmissão adequado de uma forma mais rápida, pelo facto de se descartarem alguns pacotes e assim se manter o tamanho das filas e o tempo de latência em valores baixos. A probabilidade de um pacote de um fluxo ser descartado é dependente da utilização da largura de banda que o fluxo utiliza e não do número de pacotes que transmite.

O algoritmo permite configurar três parâmetros: o tamanho mínimo da fila antes que inicie o processo de descarte aleatório, o tamanho máximo de pacotes que podem estar na fila e o número de pacotes que podem ser enviados de uma vez.

O RED é um bom algoritmo para redes de core, onde é difícil manter um algoritmo de gestão de filas de espera justo, devido à complexidade da informação necessária a manter informação de cada sessão.

3.3 GESTÃO

A gestão de redes do ponto de vista dos interesses desta dissertação comporta basicamente dois subtipos: a gestão de utilizadores e a gestão de QdS. No primeiro, gere-se a informação acerca do utilizadores, dos seus perfis e dos direitos que eles têm de acesso a recursos da rede; no segundo, gere-se a forma como os recursos são utilizados e o seu estado de sobrecarga de utilização. São esses dois assuntos que serão discutidos nas próximas secções.

3.3.1 Gestão de Utilizadores

3.3.1.1 AAAC

A maior parte dos serviços actualmente oferecidos em IP são considerados como tendo pouco valor acrescentado. Um utilizador obtém acesso à Internet gratuitamente, tendo somente que pagar uma chamada local, obtendo um serviço semelhante ao de todos os outros utilizadores ligados nesse momento. Com a crescente diversidade de serviços é necessário um controlo dos utilizadores, de modo a que a diferenciação dos serviços esteja associada a uma diversidade nos preços dos serviços prestados.

De modo a poder recolher informação acerca dos utilizadores que estão autorizados, é recolhida informação de registo de utilização que posteriormente é processada e analisada e que permite detectar utilizações abusivas e fazer contabilização. Todos estes serviços requerem coordenação entre os vários domínios administrativos dos fornecedores de acesso, em parcerias mútuas.

O objectivo fundamental de AAAC (Autenticação, Autorização, Contabilização e Tarifação) é corresponder a todos estes desafios de uma forma simples e escalável. O termo AAAC usa-se frequentemente para referenciar a plataforma para coordenar estas acções individuais através de múltiplas plataformas e tecnologias.

3.3.1.1.1 Autenticação

Autenticação é o acto de verificar uma identidade anunciada por um elemento sob a forma de uma etiqueta pré acordada [28] entre ele e a entidade de autenticação.

Autenticação define o primeiro "A" do acrónimo AAAC. Envolve a validação dos utilizadores finais, permitindo-lhes assim o acesso à rede. O processo assenta no pressuposto de que o utilizador possui uma quantidade de informação (e.g. combinação de nome de utilizador-senha, chave secreta, ou até alguma informação biométrica como impressões digitais) que serve como credencial de identificação. O servidor de AAAC compara a informação de autenticação fornecida pelo utilizador final com a que possui associada a esse utilizador na sua base de dados. Se as credencias corresponderem à informação do utilizador é-lhe garantido o acesso à rede. Um erro na informação fornecida ao servidor de AAAC resulta numa falha no processo de autenticação, causando uma negação do acesso à rede.

3.3.1.1.2 Autorização

Autorização significa o acto de determinar os direitos de acesso aos recursos que devem ser fornecidos a um utilizador [28].

Autorização define o segundo "A" do acrónimo AAAC. O processo de autorização define que direitos e serviços são permitidos ao utilizador, depois de o processo de autenticação estar concluído com êxito. Este processo pode incluir o fornecimento de um endereço IP, criação de um filtro que limite as aplicações ou protocolos a ser usados, etc.. Os processos de autenticação e de autorização são frequentemente executados em conjunto num ambiente de AAAC. Findo o processo de Autenticação, o servidor de AAAC fornece imediatamente as permissões desse utilizador.

3.3.1.1.3 Contabilização

Contabilização é o acto de recolher informação acerca da utilização de recursos, com o objectivo de análise posterior para tarifação ou atribuição de custos[28].

Contabilização - *accounting* em inglês - o terceiro "A" de AAAC, fornece a metodologia de recolha de informação acerca do consumo de recursos por um utilizador final. A informação recolhida pode ser processada e analisada para se proceder à tarifação, detectar uso abusivo de recursos, fazer planeamento da capacidade de recursos, ou até ser utilizada na resolução de questões judiciais.

3.3.1.1.4 Tarifação

Tarifação - *charging* em inglês- o "C" de AAAC, corresponde a um processo de análise dos registos de utilização, com vista à produção de um documento de facturação. Os registos de utilização produzidos durante o processo de contabilização podem ser quantidades de tráfego ou tempo de utilização dos recursos. Durante o processo de tarifação são atribuídos custos a cada unidade de unidade de utilização, que somados, definem o custo total do tráfego analisado. Os custos de utilização são definidos pelo tarifário negociado entre o cliente final e o operador do serviço. No fim do processo de tarifação é produzido um documento de facturação com vista ao pagamento do serviço prestado.

3.3.1.1.5 Funcionamento de um sistema de AAAC

A Figura 22 ilustra os componentes elementares de um sistema de AAAC. O servidor ou servidores de AAAC estão ligados à rede e funcionam como local de armazenamento central e de controlo de AAAC. O equipamento a funcionar como ponto de entrada na rede é tipicamente um *Network Access Server* (NAS), apesar de poder ser também um router ou um terminal que contenha um cliente de AAAC. Um cliente AAAC (NAS) de um ponto de acesso (POP) de uma rede comunica com o servidor AAAC, de modo a fornecer os serviços de AAAC.

O funcionamento de um sistema de AAAC pode ser descrito pelos seguintes procedimentos:

- O utilizador final liga-se a um ponto de acesso e solicita acesso à rede;
- O cliente de AAAC do NAS recolhe informação sobre as credenciais apresentadas pelo utilizador e envia-a para o servidor de AAAC;
- O servidor de AAAC processa essa informação e toma uma decisão acerca da aceitação do pedido de acesso à rede. Nesse momento, no caso de aceitação, envia para o cliente de AAAC todos os dados relevantes acerca do utilizador. No caso de negação, o servidor de AAAC envia uma mensagem ao cliente a negar o pedido;

- O cliente AAAC notifica o utilizador do sucesso do processo de registo;
- O cliente envia ao servidor de AAAC uma mensagem de contabilização, tanto durante o estabelecimento da ligação, como no seu final, de modo a que este inicie a sessão de AAAC.

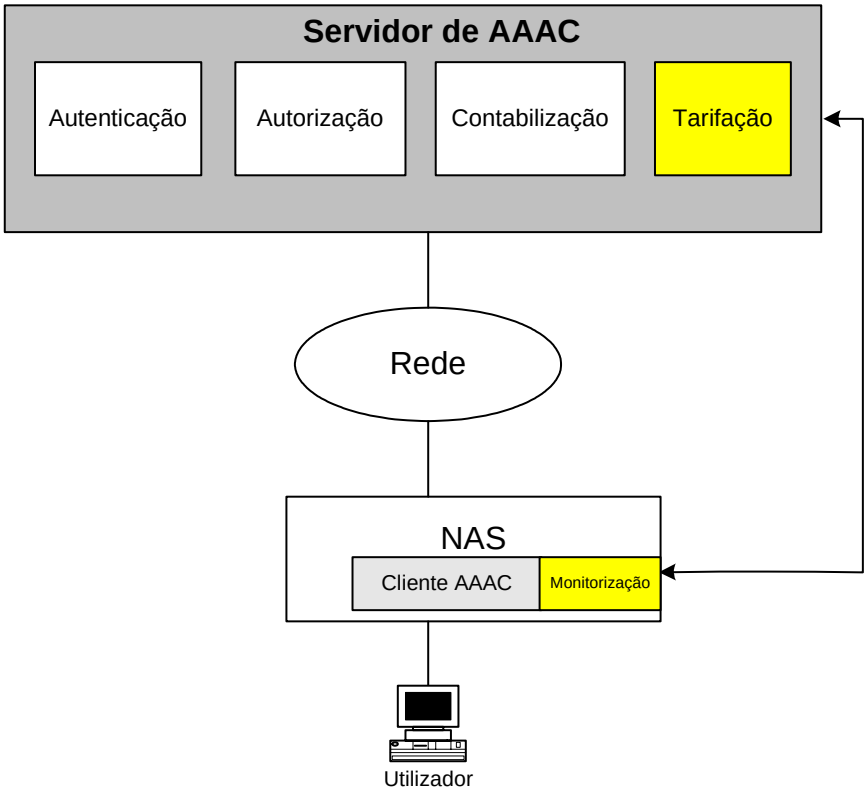

Figura 22 - Arquitectura de um sistema de AAAC

Um servidor de AAAC pode actuar como um ponto de controlo de acesso administrativo central para clientes de AAAC que contenham tecnologia proprietária dos fabricantes. Permite ainda que as capacidades de AAAC sejam adicionadas aos servidores de AAAC e posteriormente aos clientes, sem que o serviço das funcionalidades disponíveis tenha que ser para isso descontinuado.

A vantagem deste tipo de arquitectura cliente-servidor é que o servidor de AAAC pode ser hospedado numa máquina normal, o que em termos de relação qualidade/preço é sempre vantajoso. Ao mesmo tempo também permite que um grande volume de disco seja destinado ao alojamento da informação de AAAC e uma administração optimizada e flexível da base de dados. Permite ainda aos fornecedores de serviço de acesso ter capacidade de resposta às grandes quantidades de pedidos de AAAC provenientes dos seus servidores de NAS, mantendo capacidade para guardar a informação de contabilização de cada uma das ligações dos utilizadores finais.

Foram desenvolvidos pelo IETF dois protocolos de comunicação entre o NAS e o servidor de AAAC: o RADIUS, de vasta utilização, e o Diameter, seu sucessor.

3.3.1.1.6 RADIUS

O RADIUS foi desenvolvido em meados da década de 90 pela *Livingston Enterprises* (mais tarde adquirida pela *Lucent Technologies*) de modo a dotar o seu equipamento de NAS de serviços de AAAC. O IETF aceitou o trabalho da *Livingston* em 1996 e criou o RADIUS WG (Working Group) do qual resultaram o formato e as funções básicas que foram tornadas norma na RFC 2138.

Os princípios de funcionamento do RADIUS são:

- ***Funcionamento baseado na arquitectura Cliente/Servidor***: Um cliente RADIUS é colocado no servidor de NAS e comunica via rede com o servidor RADIUS que funciona noutra máquina. Além disso um servidor RADIUS pode servir de cliente *proxy* para outro servidor do mesmo tipo, ou outro servidor de autenticação.
- ***Segurança na rede***: Todas as comunicações entre um cliente RADIUS e o servidor são autenticadas através do uso de uma chave partilhada que nunca é enviada através da rede. Para além disso as *passwords* de utilizador contidas nas mensagens RADIUS são encriptadas para evitar que os *hackers* as leiam se capturarem o tráfego da rede.
- ***Autenticação flexível***: O RADIUS pode suportar múltiplos mecanismos de autenticação, incluindo *Password Authentication Protocol* (PAP), *Challenge Handshake Authentication Protocol* (CHAP) ou *Extended Authentication Protocol* (EAP).
- ***Pares atributo-valor (Attribute Value Pairs - AVP's)***: As mensagens de RADIUS transportam a informação de AAAC codificada em campos predefinidos chamados atributos (ou pares atributo-valor). Exemplos de AVP's incluem o nome do utilizador, a *password*, o protocolo, endereço IP para o utilizador, etc.

O RFC 2138 [25] define uma lista mais completa de AVP's RADIUS suportados pelos clientes e pelos servidores.

O Funcionamento do RADIUS pode ser descrito da seguinte forma (Figura 23):

1. Inicialmente o cliente RADIUS do NAS envia uma mensagem de pedido de acesso ao servidor RADIUS com as credenciais do utilizador. Usa o protocolo *User Datagram Protocol* (UDP) para enviar a informação numa mensagem encriptada através da rede até ao servidor RADIUS.
2. O servidor RADIUS procura na sua base de dados um registo que contenha a identificação apresentada no AVP *User-Name*.

3. Se existir um registo na base de dados do servidor e se a senha encontrada estiver correcta o servidor RADIUS devolve uma mensagem de *Access-Accept* ao servidor NAS indicando a aceitação ao cliente. É ainda enviada alguma informação de configuração necessária para completar a ligação, como o endereço IP para o utilizador final, ou um filtro que limite a ligação a um tipo de protocolo, como Telnet ou HTTP. A mensagem pode ainda conter alguma informação extra como a identificação do servidor de NAS ou o seu endereço.
4. Se não encontrar, o servidor devolve uma mensagem de *Access-Reject* para o NAS que pode opcionalmente criar uma mensagem de texto indicando a razão da falha notificando o utilizador.

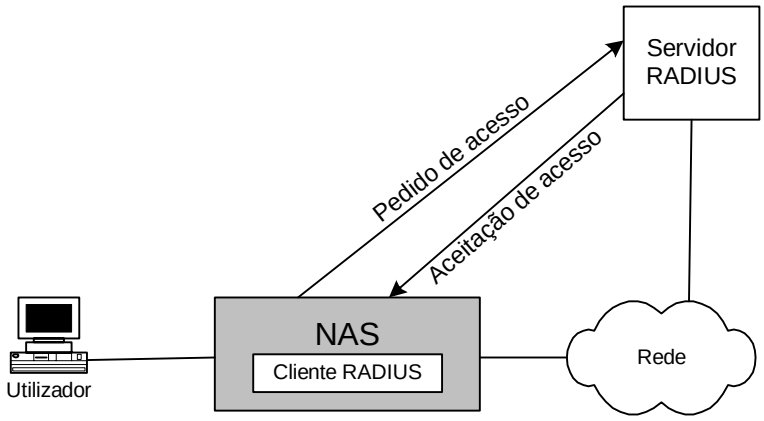

Figura 23 - Funcionamento do RADIUS

3.3.1.1.6.1 Pacote do RADIUS

A informação RADIUS é enviada em pacotes RADIUS, e consiste num cabeçalho (ver Figura 24) e objectos RADIUS. Esses objectos são os de AVP's, e por sua vez também contém cabeçalho e informação.

Figura 24 - Formato do pacote RADIUS

O cabeçalho de um pacote RADIUS contém quatro campos diferentes:

- ***Código***: um campo de 8 bits que determina o tipo de pacote em questão;
- ***Identificador***: campo de 8 bits que serve como identificador dos pedidos e que ajuda a fazer a ligação entre os pedidos e as respostas;
- ***Comprimento***: campo de 16 bits que define o tamanho do pacote RADIUS, incluindo todos os AVP's;
- ***Autenticador***: campo de 32 bits usado para autenticar a resposta do servidor RADIUS e também para definir o algoritmo que encripta as identificações e as senhas dos utilizadores nas mensagens entre o cliente de RADIUS e o servidor.

Os AVP's RADIUS contém informação específica de autenticação, de autorização, de Contabilização e de configuração. Os AVP's consistem em 3 campos como se pode ver na Figura 25.

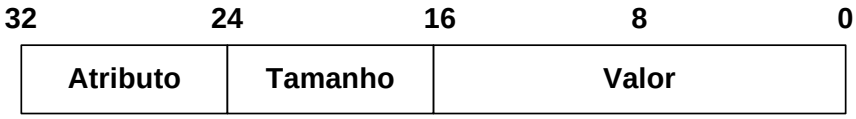

Figura 25 - Formato de um AVP

3.3.1.1.7 Diameter

O protocolo RADIUS é desde algum tempo a esta parte usado em serviços AAAC para ligações ponto-a-ponto (PPP) e para acessos a terminais remotos, mas à medida que os routers e servidores de NAS cresceram em tamanho e em quantidade o protocolo mostrou-se pouco útil para redes de grande dimensão. O protocolo Diameter não foi desenvolvido do zero, mas mantendo o formato básico do RADIUS, e tentando eliminar algumas deficiências já conhecidas do RADIUS. Apesar das alterações introduzidas neste novo protocolo, o Diameter mantém a compatibilidade com o RADIUS em vários pontos da sua arquitectura (no formato dos AVP's, na numeração dos *command-code)*. Isto permite aos servidores Diameter facilmente mapearem os atributos RADIUS para os AVP's Diameter.

A ideia que esteve na base do Diameter foi a de criar um protocolo base que pudesse ser genericamente estendido a novos métodos de acesso. O desenvolvimento actual do Diameter é no

sentido de se limitar o alcance do protocolo ao acesso à Internet através ligações comuns PPP, e por critérios definidos pelos modelos ROAMOPS (*roaming operations*) e Mobile-IP.

3.3.1.1.7.1 Arquitectura do Diameter

O Diameter consiste num protocolo base [29] com diferentes extensões e aplicações como *CMS-Security* [31], Network Access Server REQuirements (*NASREQ)* [33] e *Mobile-IP* [32] como ilustra a Figura 26. As funcionalidades básicas comuns a todas as aplicações e serviços são implementadas no protocolo base, enquanto que todas as funcionalidades específicas existem dentro das extensões.

O protocolo base [29] dispõe de capacidade de negociação da forma como as mensagens são enviadas. Define também regras que são aplicadas a todas as trocas de mensagens entre nós Diameter. O protocolo Diameter base especifica o formato e o transporte das mensagens. Procede ao reporte de erros e define serviços de segurança para serem utilizados por todas as aplicações e que devem ser suportados por todas as implementações Diameter.

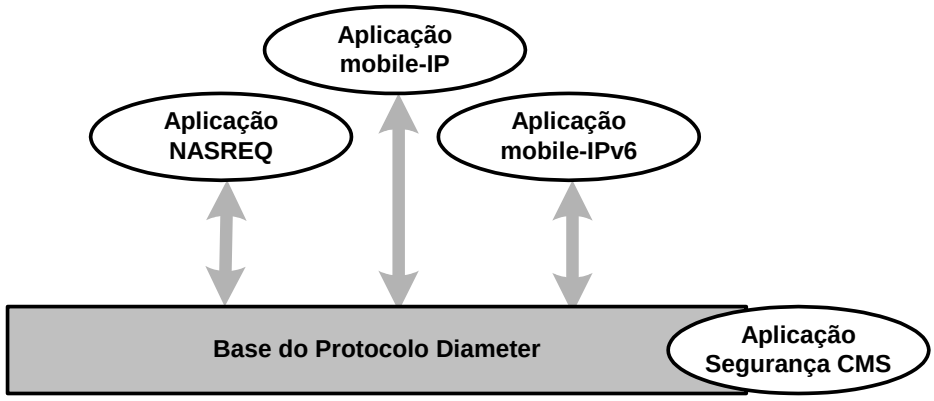

Figura 26 - Arquitectura do Protocolo Diameter

A Figura 26 mostra um esquema da arquitectura do protocolo Diameter. O protocolo base está intimamente ligado ao módulo de segurança CMS de modo a fornecer segurança a todas as aplicações. Pode-se verificar ainda da Figura 26 que os servidores com outros módulos com diferentes funcionalidades necessitam de usar também o protocolo base.

3.3.1.1.7.2 Mensagens Diameter

As mensagens Diameter, tal como as mensagens RADIUS, consistem num cabeçalho seguido de um conjunto de AVP's.

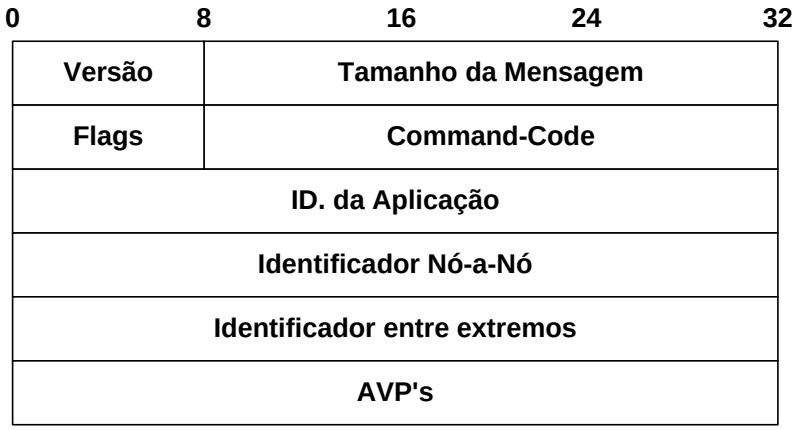

Figura 27 - Cabeçalho do Diameter

O cabeçalho Diameter contém os seguintes campos:

- ***Versão***: campo de oito bits que indica a versão do protocolo;
- ***Tamanho***: campo de três octetos que indica o tamanho da mensagem, incluindo o cabeçalho;
- ***Flags***: campo de oito bits:
  - ***R***: quando activo indica que a mensagem é um pedido, caso contrário uma resposta;
  - ***P***: no caso de estar activo indica que a mensagem pode ser reencaminhada; caso contrário deve ser processada localmente;
  - ***E***: indica que a mensagem é uma mensagem de erro;
  - ***T***: flag activa após uma falha numa ligação com o objectivo de evitar mensagens replicadas;
  - Bits de reserva: quatro bits reservados para uso futuro;
- ***Comand-Code***: campo de três octetos usado para definir o comando da mensagem. Os valores deste campo são definidos pelos IANA;
- ***ID. da aplicação***: campo de quatro octetos que é usado para identificar a que aplicações a mensagem é aplicável. A identificação da aplicação presente no cabeçalho do pacote Diameter deve ser a mesma presente nos AVP's dessa mensagem;
- ***Identificador Nó-a-No:*** campo de quatro octetos que serve para ajudar a determinar quais as respostas a cada pedido;
- ***Indetificador entre extremos:*** campo de quatro octetos que serve para ajudar a detectar mensagens duplicadas;
- ***AVP's:*** conjunto de AVP's de cada mensagem Diameter.

Os AVP's consistem num cabeçalho seguido de dados ligados ao AVP em questão.

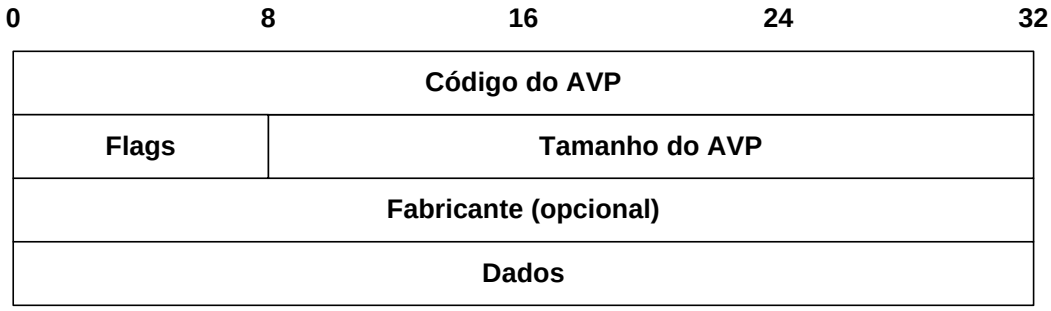

Figura 28 – Formato de um AVP

Um AVP contém os seguintes campos:

- ***Código***: campo que identifica de forma única o atributo. Os primeiros 256 números de AVP's estão reservados para compatibilidade com RADIUS. O valor dos códigos é definido pelo IANA;
- ***Flags do AVP:*** as *flags* do AVP indicam ao servidor como cada atributo deve ser tratado. As extensões do Diameter podem definir valores a serem usados dentro dos AVP's:
  - ***V***: indica que as mensagens são específicas do fabricante. Caso esteja activa indica a presença do campo Fabricante no cabeçalho do AVP e que o código pertence aos códigos proprietários do fabricante;
  - ***M***: representa o bit mandatório e indica que quem recebeu a mensagem a deve suportar, caso contrário deve rejeita-la. No caso das mensagens sem o bit M activo tem um caracter somente informatório, e nesse caso pode ser ignorada tal mensagem;
  - ***P***: bit que indica a necessidade de encriptação para a segurança fim-a-fim;
  - ***Bits reservados***: seis bits reservados para uso futuro;
- ***Tamanho***: campo de dois octetos que define o tamanho do AVP incluindo o cabeçalho;
- ***Fabricante***: campo opcional de quatro octetos identificativo do fabricante que é atribuído pelo IANA;
- ***Dados***: campo de vários octetos que recebe a informação do atributo.

3.3.2 Gestão de QdS a nível de rede

3.3.2.1 Gestão Baseada em Políticas

A complexidade crescente das redes de comunicações tornam necessários métodos mais eficientes de gestão. A configuração individual dos elementos de rede, além de ser muito trabalhosa e de despender muito tempo, é muito propícia a erros. Além disso os erros de configuração dos nós de rede são por vezes difíceis de detectar e têm consequências muito nefastas para o tráfego na rede. O conceito de Gestão Baseada em Políticas [16] propõe uma solução para este problema através da gestão e controlo da rede por uma entidade de nível superior que se encarrega de configurar os elementos de rede. O administrador especifica um conjunto de políticas no Gestor de Baseado em Políticas (GBP) que são válidas para todo o domínio, independentemente das características de cada nó de rede em particular. O GBP encarrega-se de distribuir os políticas para todos os nós de rede e de traduzir as políticas definidas genericamente para cada um dos nós. A distribuição da configuração dos elementos de rede por uma maquina central permite a eliminação dos erros de configuração, bem como a garantia de consistência na configuração de toda a rede. Estes duas características são requisitos essenciais na configuração de uma rede com garantias de QdS.

A gestão baseada em políticas traz entre outras as seguintes vantagens:

- Possibilidade de criação de um camada de abstracção possibilitando que a configuração do GBP possa ser feita por quem não domine a sintaxe de configuração dos routers;
- Permite ter garantias de configuração de forma consistente todos os elementos da rede;
- Elimina o erro de configuração nos routers devido ao erro humano;

A arquitectura da plataforma de gestão baseada em políticas desenvolvida pelo IETF [17] descreve os componentes principais (Figura 29), apesar de não definir nenhum pormenor de implementação ou linguagem. Entre esse componentes temos:

- ***Policy Decision Point (PDP)***: ponto de decisão de políticas é o ponto onde as decisões são tomadas. É também designado por servidor de políticas;
- ***Policy Enforcement Point (PEP)****:* ponto da rede onde as decisões são aplicadas é responsável por executar as acções definidas pelo PDP*;*
- ***Policy Transfer Protocol***: é usado para transportar a informação de policiamento entre o PDP e o PEP. O *Common Open Policy Service* (COPS) é um protocolo

proposto para transferência de políticas, mas existem outros protocolos como RSVP, o RADIUS e Diameter que dispõem de suporte para esse efeito.

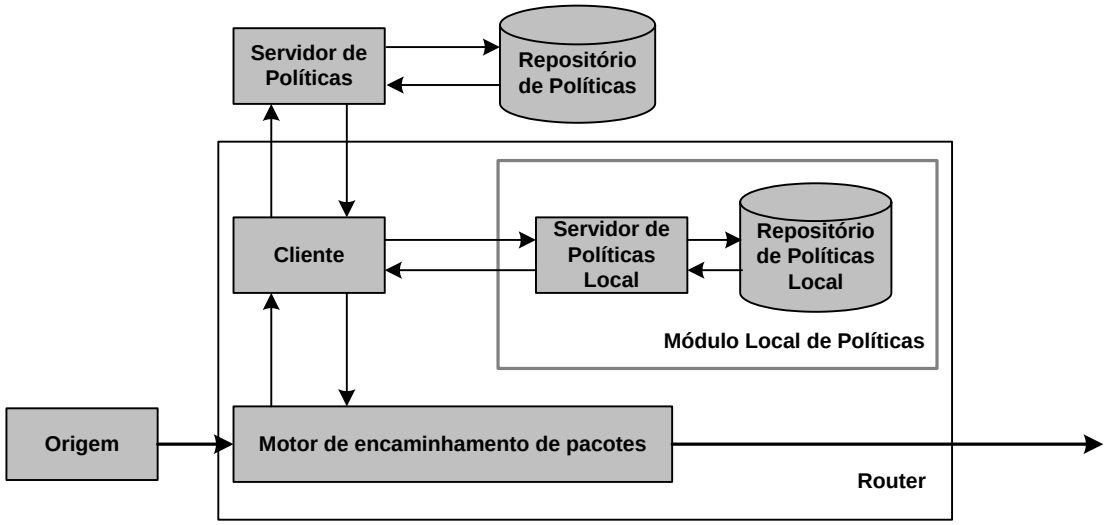

Figura 29 - Modelo de implementação de Políticas

3.3.2.2 Common Open Policy Service (COPS)

O protocolo *Common Open Policy Service* (COPS) definido na RFC 2748 [34] é o protocolo que foi especialmente definido para fazer transporte de informação de políticas entre o PDP e os PEP. O modelo básico do protocolo COPS está esquematizado na Figura 30.

Os recursos são alocados ou libertados dentro de um nó por um PEP. Um nó pode não implementar as políticas, no caso de um nó que não suporta COPS (nó 2). Os nós da rede são agrupados em domínios administrativos, onde existe sempre pelo menos um servidor de políticas. Dentro de um servidor de políticas existe sempre um PDP, que toma as decisões acerca da gestão dos recursos. Pode ainda existir um PDP local em cada nó de rede.

O protocolo COPS é baseado na arquitectura cliente/servidor onde o PEP envia pedidos, actualizações e remove reservas ao PDP, e o PDP devolve decisões ao PEP. Se um PDP remoto não está disponível, por exemplo devido ao um erro da rede, o PEP deve tentar ligar-se a um servidor de PDP de reserva ou funcionar com base no PDP local (LPDP). Contudo assim que a ligação com o PDP remoto é recuperada, o PEP deve actualizar o PDP com as decisões que foram tomadas na sua ausência. A ligação entre PEP e PDP remoto é fiável, uma vez que o protocolo COPS usa TCP como protocolo de transporte. As decisões são sempre tomadas pelo PDP, apesar do PEP poder tomar decisões recorrendo ao seu LPDP, a decisão final cabe sempre ao PDP em condições normais de funcionamento. É no entanto possível que o PEP tome uma decisão com base

no seu PDP local, sem que consulte o PDP previamente. Este tipo de acção só se processa no caso de indisponibilidade do PDP e as decisões tomadas localmente devem ser anunciadas ao PDP remoto assim que isso seja possível, e podem ser revogadas por este se assim o considerar.

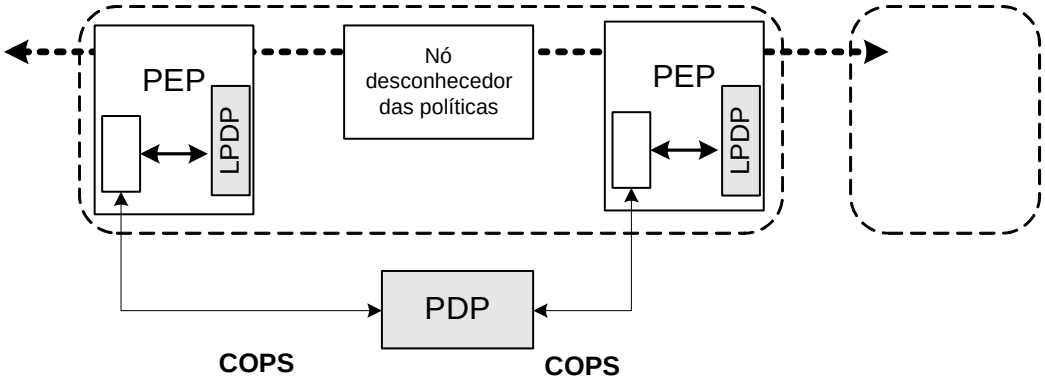

Figura 30 - Modelo básico do COPS

O PEP inicia a ligação TCP e depois disso a comunicação entre eles consiste num conjunto de trocas de pergunta e reposta. O protocolo COPS é *stateful*:

- O estado dos pedidos/decisões é partilhado entre o cliente e o servidor. Os pedidos do PEP cliente são guardados no PDP até que sejam explicitamente apagados pelo PEP.
- O estado dos vários eventos (pares pedido/resposta) podem ser associados. Isto significa que o servidor pode responder a novos pedidos de uma forma diferente por causa de pares cliente/resposta anteriormente instalados que estejam relacionados.
- Os PDP podem também enviar mensagens não solicitadas para o PEP, por exemplo podem alterar no PEP decisões anteriormente tomadas. O PEP pode enviar informação de contabilização para o PDP.

3.3.2.2.1 COPS-RSVP

O COPS é um protocolo especialmente concebido para comunicar informação de políticas a dispositivos de rede. Tem sido desenvolvido pelo *RSVP Admission Policy Working Group (RAP WG)* do IETF, especialmente como meio de fornecimento de mecanismos de controlo de admissão baseados em políticas a pedidos de recursos de redes.

O RSVP é um protocolo de sinalização que tem como objectivo a reserva de recursos de rede para garantir que as aplicações possam transmitir dados entre uma origem e um destino com uma determinada velocidade e qualidade garantidas.

Combinando o COPS e o RSVP consegue-se uma gestão centralizada da rede, usando o COPS como forma de reencaminhar as decisões de políticas de controlo dos clientes (PEP's) para os servidores de políticas (PDP's). COPS-RSVP [37] suporta então as seguintes funcionalidades:

- **controlo de admissão**: A reserva RSVP é aceite ou rejeitada tendo como base de decisão o estado de sobrecarga dos elementos de rede existentes entre extremos; também pode considerar quaisquer políticas presentes no PDP, e.g. controlo de acesso por IP origem, PEP de acesso;
- **garantia de QdS**: A reserva RSVP, no caso de aceite, vai garantir que os recursos requisitados vão ser reservados para a utilização do fluxo que os requisitou, e que essa reserva vai ser mantida enquanto o pedido feito estiver válido;
- **classificação dos dados**: enquanto a reserva se encontra válida, os pacotes que pertencem a esses fluxo RSVP são separados do restante tráfego e enviados como fazendo parte do fluxo reservado;
- **policiamento dos dados**: os pacotes pertencentes a um fluxo RSVP que excedam a largura de banda reservada são marcados com um precedência mais baixa, podendo em certas circunstancias ser até descartados.

3.3.2.2.1.1 Funcionamento do COPS-RSVP

É possível configurar um router para processar todas as mensagens RSVP que lhe chegam de acordo com um conjunto de políticas presentes num servidor de políticas, como ilustra a Figura 31.

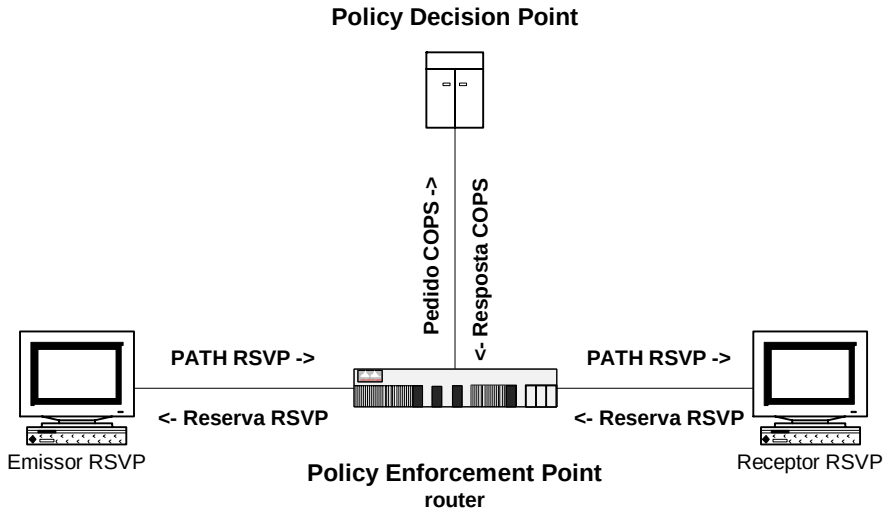

Figura 31 - Exemplo de COPS-RSVP

O funcionamento de uma rede como a descrita na figura pode resumir-se nos seguintes passos:

- Assim que o router (PEP) é iniciado, tenta-se ligar ao PDP que existe na sua configuração;
- Assim que uma mensagem de sinalização RSVP chega a um PEP, este pergunta ao servidor como processar a mensagem;
- O servidor, depois de analisar o pedido e consultar a sua lista de políticas, toma uma decisão e responde ao PEP.
- O PEP ao receber a resposta implementa a decisão enviada pelo PDP.

3.3.2.2.2 COPS-PR

O modelo de delegação de decisões em que se baseia o COPS-RSVP corresponde a um conjunto de eventos em que um PEP precisa de uma resposta do PDP instantânea, sob a forma de uma decisão baseada numa política. Neste cenário de delegação de decisões, o PEP delega a responsabilidade das decisões a um PDP externo.

Por outro lado o modelo de configuração ou de aprovisionamento [38], não mantém o modelo exclusivo de pergunta/resposta. Permite que o PDP tome a iniciativa de face a um evento, e.g. mudança de configuração ou alteração das condições da rede, enviar mensagens de configuração ao PEP. O aprovisionamento pode ser feito através de uma configuração completa do PEP, e.g. uma configuração completa de um router, ou então através de configurações parciais como uma alteração dos parâmetros de um filtro de marcação *DiffServ*.

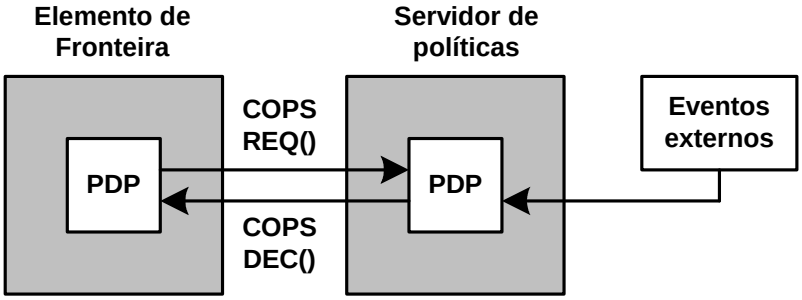

Figura 32 - Modelo de COPS-PR

O servidor de políticas do COPS-PR descreve ao PEP os seus parâmetros configuráveis. Se existir uma mudança nestes parâmetros básicos é enviado um pedido de actualização. As decisões não são agora só enviadas em resposta aos pedidos, mas também sempre que o PDP necessite de responder a eventos externos, como alterações de políticas ou SLA.

Quando um dispositivo arranca, abre uma ligação COPS com o PDP primário. Quando a ligação é estabelecida o PEP envia informação acerca de si próprio ao PDP sob a forma de um pedido de configuração. Esta informação inclui informação específica do cliente como tipo de hardware, versão de software, e informação de configuração. Durante esta fase o cliente pode também especificar o tamanho máximo das mensagens suportadas.

Como resposta o PDP procura todas as informações de políticas relevantes para este cliente. Ao receber as políticas de aprovisionamento, o cliente mapeia-as nos seus mecanismos de QdS e instala-as. Se as condições mudarem no PDP, por exemplo se o PDP detectar que houve uma alteração nas políticas, envia ao PEP somente as alterações a fazer à configuração anteriormente enviada que o PEP altera localmente.

Se a configuração do cliente for alterada de forma a que informação relevante para o PDP tenha mudado, e.g. mudança de hardware, ou instalação de software, o PEP deve assincronamente enviar essa informação para o PDP sob a forma de um pedido de actualização de configuração. Ao receber esta nova informação o PDP envia ao PEP informação actualizando a configuração presente no PEP.

Este modelo é baseado no conceito de *Policy Information Bases (*PIB*)* que definem os dados das políticas.

3.3.2.2.2.1 Policy Information Base

A informação transportada pelo COPS-PR é um conjunto de dados de políticas. O protocolo define uma estrutura de dados chamada de PIB, que identifica o tipo e propósito de uma

informação de política que é despejada pelo PDP no PEP, ou enviada pelo PEP ao PDP como notificação.

O conjunto de nomes de uma PIB é comum ao PEP e ao PDP e as instancias pertencentes a esse conjunto são únicas dentro do contexto definido por um *Client-Type*, uma ligação TCP entre um PEP e um PDP e um *Request-State*.

A PIB pode ser descrita como uma arvore conceptual onde os ramos da arvore representam estruturas de dados ou classes de aprovisionamento - *Provisioning Class* (PRC) - , enquanto que as folhas representam Instâncias de Aprovisionamento - *Provisioning Instance* (PRI). Podem existir várias instancias de dados PRI para uma data estrutura de dados PRC. Por exemplo se alguém quer instalar vários filtros de controle de acesso, a PRC pode representar um filtro de controlo de acesso e cada PRI pode representar uma instancia de filtros de controlo de acesso a ser aplicado. A arvore pode-se representar como na Figura 33.

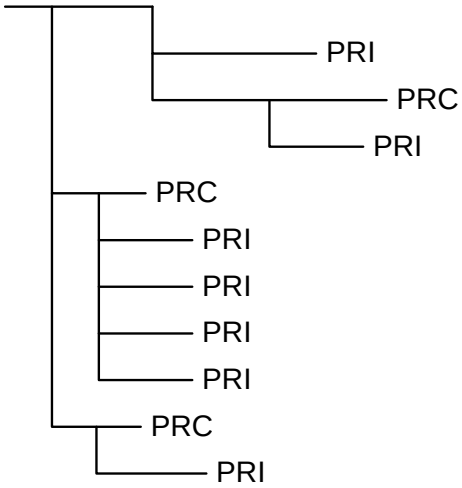

Figura 33 - A árvore de uma PIB

Instancias das classes de políticas PRI são identificadas por um *Provisioning Instance Identifier* (PRID) único. Um PRID é um identificador que é transportado por um objecto COPS <Named ClientSI> ou <Named Decision Data> , que identifica uma instancia particular de uma classe.

3.3.2.2.2.2 Vantagens do COPS-PR

O COPS-PR foi desenvolvido para criar uma plataforma optimizada para aprovisionamento eficiente de políticas entre dispositivos baseados no requisitos definidos em [39], tais como:

- independência das políticas a implementar;

- independência do protocolo RSVP;
- possibilidade de remover decisões anteriormente instaladas;
- flexibilidade nas políticas a implementar;
- suporte para contabilização e monitorização;
- suporte para nós que não reconheçam as políticas:
- escalabilidade;
- segurança e prevenção de *Denial-Of-Service*.

O COPS-PR permite um transporte eficientes de atributos, trocas de grandes quantidades atómicas de dados, e reportar de uma forma flexível e eficiente os dados. Utilizando um mecanismo de protecção evita que múltiplos servidores estejam a configurar um cliente de uma determinada área de controlo. A configuração é vedada até à consola local de configuração enquanto um PEP está ligado a um PDP via COPS. O COPS usa um transporte fiável baseado em TCP com partilha de estado entre o PDP e o PEP, permitindo assim que se façam somente acções de configuração parciais. Mesmo que o cliente ou o servidor sejam reiniciados, o outro vai descobri-lo em pouco tempo.

3.3.2.2.3 A estrutura do COPS

O protocolo foi concebido de maneira a que as mensagens contêm objectos que se auto-identificam. Os elementos de policiamento são pedaços de informação necessários para aplicação das regras de policiamento. Os objectos são independentes do protocolo de sinalização de QdS que é usado. As mensagens COPS estão divididas em cabeçalho e em objectos.

3.3.2.2.4 O cabeçalho COPS

Uma mensagem COPS começa sempre com um cabeçalho comum (Figura 34).

0		1	2	3
Versão	Flags	Op. Code	Client-Type	
Tamanho				

Figura 34 - Cabeçalho comum COPS

Os campos do cabeçalho são os seguintes:

- **Versão**: campo de 4 bits que indica a versão do protocolo COPS. A versão corrente é a 1;
- **Flags**: Bit-Field de 4 bits. Este campo tem apenas definido o bit 0x1 *Solicited Message Flag Bit* que se encontra activo quando uma mensagem é solicitada por outra mensagem COPS;

- **Op. Code**: campo de 8 bits que indica qual a operação COPS que podem ser:
  - *Request* (REQ): mensagem enviada sob a forma de um pedido;
  - *Decision* (DEC): mensagem de resposta, normalmente enviada sob a forma de decisão a pedido anteriormente enviado;
  - *Report State* (RPT): mensagem utilizada pelo PEP para comunicar ao PDP o seu sucesso ou a sua falha, na execução da decisão do PDP;
  - *Delete Request State* (DRQ): pedido de anulação de decisão anterior
  - *Synchronize State Req* (SSQ): mensagem enviada quando o PDP quiser saber o estado completo do PEP;
  - *Client-Open* (OPN): mensagem enviada pelo PEP pedindo o estabelecimento de sessão; *Client-Accept* (CAT): mensagem enviada pelo PDP anunciando que a ligação com PEP foi aceite;
  - *Client-Close* (CC): mensagem enviada pedindo fecho sessão COPS;
  - *Keep-Alive* (KA): mensagem enviado periodicamente entre PDP e PEP anunciando que estão activos;
  - *Synchronize Complete* (SSC): mensagem anunciando o fim de período de sincronismo de estados;
- **Client-type**: campo de 16 bits que identifica o cliente de policiamento. A interpretação dos objectos encapsulados nas mensagens é dependente do tipo de cliente; por exemplo as mensagens de KA devem ter um client-type de 0.
- **Length**: campo de 32 bits que indica em bytes o tamanho da mensagem. O tamanho inclui o cabeçalho e todos os objectos.

As mensagens COPS devem ser alinhadas em intervalos de 4 octetos.

3.3.2.2.5 Formatos dos objectos COPS

O cabeçalho COPS define o tipo de mensagem. Ao cabeçalho seguem-se um numero de elementos que são usados para que se tomem decisões e para comunicar decisões anteriormente tomadas. O conteúdo dos elementos de policiamento dependem do protocolo de sinalização de COPS, mas a estrutura base do protocolo COPS mantém-se intacta. Os elementos de policiamento que são definidos pelo protocolo têm uma estrutura definida na Figura 35:

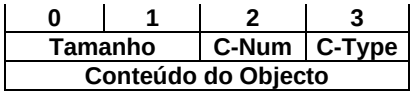

Figura 35 - Formato de um objecto COPS

O campo *Tamanho*, que tem dois bytes, indica o tamanho, incluindo o cabeçalho, da informação que compõe o objecto. Se o tamanho do objecto não for múltiplo de 32 bit são acrescentados zeros ao fim do mesmo, de modo a que o objecto fique alinhado. No receptor, os zeros colocados para alinhar o objecto são retirados utilizando o valor do campo *Tamanho* e assim determinar qual a informação útil presente no objecto.

Tipicamente o campo *C-Num* (8 bits) identifica o tipo de informação contida no objecto, e *C-Type* identifica o subtipo ou a versão do mesmo. Por exemplo a interface de entrada é sinalizada com um *C-Num* de 3 e a interface de saída é sinalizada com *C-Num* de 4. Nestes dois casos um *C-Type* de 1 significa um endereço IPv4 enquanto que um *C-Type* de 2 indica um endereço IPv6.

Os valores possíveis do campo *C-Num* e os seus significados são apresentados na Tabela 2:

C-Num	***Nome***	***Explicação***
1	*Handle*	O objecto *Handle* encapsula um valor único que identifica uma acção . A maior parte das operações COPS usa esta identificação.
2	*Context*	Especifica o tipo de evento(s) que despoletaram a *query*. É obrigatório nas mensagens de pedido.
3	*In Interface*	Este objecto é usado para identificar a interface de entrada a que um determinado pedido se aplica e o endereço de onde a mensagem recebida foi originada.
4	*Out Interface*	Este objecto é usado para identificar o interface de saída a que determinado pedido se aplica e o endereço para onde a mensagem deve ser enviada.
5	*Reason Code*	Especifica a razão pela qual o *request state* foi apagado. Aparece na mensagem de *delete request* (DRQ).
6	*Decision*	Indica uma mensagem de *Decision* enviada pelo PDP. Aparece nas mensagens de resposta.
7	*LPDP Decision*	Indica uma decisão tomada pelo (LPDP). Pode aparecer nos pedidos.
8	*Error*	É usada para identificar um erro no protocolo COPS.
9	*Client Specific Info*	Indica que o conteúdo é informação específica do tipo de cliente, e.g. no caso de um cliente RSVP é usado para enviar mensagens *Path-RSVP*.
10	*Keep-Alive Timer*	Tempo de *Keep-Alive* em segundos.
11	*PEP Identification*	O objecto *PEP Identification* é usado para identificar o cliente (PEP) junto de um PDP remoto.
12	*Report Type*	Tipo de Relatório do estado do pedido para o *handle*.
13	*PDP Redirect Address*	Um PDP quando fecha uma sessão com um PEP de um determinado de tipo pode usar este objecto para redireccionar o PEP para um determinado PDP enviando-lhe um endereço e número de porto TCP.
14	*Last PDP Address*	Quando determinados PEP enviam uma mensagem de *Client-Open* devem especificar qual o último PDP com o qual estabeleceram uma ligação com sucesso (do qual receberam uma mensagem de *Client-Accept*) desde que re-arrancaram da ultima vez.
15	*Accounting Timer*	Valor do temporizador opcional usado para determinar o intervalo mínimo entre relatórios periódicos de contabilização.
16	*Message Integrity*	O objecto de integridade inclui um número sequencial usado para autenticar e validar a integridade de uma mensagem COPS.

Tabela 2 - COPS specific objects

Todos os objectos COPS tem conteúdo. Esse conteúdo, que é variável, é dependente dos campos *C-Num* e *C-Type*. As mensagens entre o PEP e o PDP são construídas com esses objectos. Seguem-se alguns exemplos de mensagens.

3.3.2.2.6 Mensagens COPS típicas

Pelo facto de as mensagens formarem um estrutura hierárquica, é bastante fácil representa-las através de uma notação *Backus Naur Form* (BNF). A colocação dos objectos nas mensagens deve ser feita por uma determinada ordem, tal como é feito nos exemplos. No caso de ser necessária autenticação ou segurança é necessário que o objecto de integridade seja sempre o ultimo objecto da mensagem. Seguem-se alguns objectos típicos.

3.3.2.2.6.1 Request (REQ) - Pedido do PEP ao PDP

O PEP define um identificador único (*handle*) que serve para que o PDP possa manter o estado desse pedido. O *handle* serve para que o PDP diferencie os pedidos, e assim possa guardar informação acerca da resposta que deu a cada um. No caso de existirem alterações locais no PEP, este é responsável por as comunicar ao PDP. O formato de uma mensagem REQ é:

```
<Request Message> ::= <Common Header>
                      <Client Handle>
                      <Context>
                      [<IN-Int>]
                      [<OUT-Int>]
                      [<ClientSI(s)>]
                      [<LPDPDecision(s)>]
                      [<Integrity>]

<ClientSI(s)> ::= <ClientSI> | <ClientSI(s)> <ClientSI>

<LPDPDecision(s)> ::= <LPDPDecision> |
                      <LPDPDecision(s)> <LPDPDecision>

<LPDPDecision> ::= [<Context>]
<LPDPDecision: Flags>
                      [<LPDPDecision: Stateless Data>]
                      [<LPDPDecision: Replacement Data>]
                      [<LPDPDecision: ClientSI Data>]
                      [<LPDPDecision: Named Data>]
```

O cabeçalho de uma mensagem REQ segue o descrito na secção 3.3.2.2.4. O objecto *Context* define o contexto em que os outros objectos devem ser interpretados: a chegada de uma mensagem, a alocação de recursos locais, o reencaminhamento de uma mensagem de saída ou informação de configuração.

O *ClientSI* contem informação específica do cliente, por exemplo informação de uma mensagem de um protocolo de sinalização de QdS. O objecto *LPDPDecision* contém informações acerca de decisões tomadas pelo PDP local e que precisam de ser validadas pelo PDP remoto.

3.3.2.2.6.2 Decision (DEC) - Decisão do PDP ao PEP

O PDP responde ao PEP enviando uma mensagem de decisão. Se o PDP não responder dentro de um determinado limite de tempo, o PEP remove o *handle*, cria um novo e tenta novamente. Segue-se a estrutura de uma mensagem de decisão:

```
<Decision Message> ::= <Common Header>
                       <Client Handle>
                       <Decision(s)> | <Error>
                       [<Integrity>]

<Decision(s)> ::= <Decision> | <Decision(s)> <Decision>

<Decision> ::= <Context>
               <Decision: Flags>
               [<Decision: Stateless Data>]
               [<Decision: Replacement Data>]
               [<Decision: ClientSI Data>]
               [<Decision: Named Data>]
```

O PDP pode enviar várias decisões para o PEP, quer por necessidade, quer em casos de comportamentos anómalos.

3.3.2.2.6.3 Client-Open (OPN) - do PEP para o PDP

O PEP usa a mensagem de *Client-Open* para comunicar ao PDP quais os tipos de *client types* que suporta. O PEP também pode especificar o ultimo PDP a que se ligou. Segue-se a estrutura de uma mensagem de *Client-Open*:

```
<Client-Open> ::= <Common Header>
                  <PEPID>
                  [<ClientSI>]
                  [<LastPDPAddr>]
                  [<Integrity>]
```

O PEPID é um nome simbólico que o identifica dentro de um domínio administrativo. A identificação é uma *string* ASCII que pode ser um endereço IP ou um nome DNS do PEP. O PEP pode ainda enviar alguma informação específica adicional para o PDP. O objecto de integridade é usado quando se pretende aumentar a segurança.

3.3.2.2.6.4 Client-Accept (CAT) - do PDP para o PEP

O PDP responde com uma mensagem *Client-Accept* se aceitar o pedido de *Client-Open*. O PDP devolve um valor de *KA Timer*, que define qual o tempo máximo do intervalo entre mensagens de *keep-alive.* Opcionalmente o PDP pode também enviar um tempo de ACCT, que define o mínimo intervalo entre mensagens de relatórios contabilização enviadas pelo PEP.

3.3.2.2.7 Integridade das Mensagens

As mensagens de *Client-Open* e de *Client-Accept* também são usadas na negociação de segurança, desde que isso seja configurado previamente. Se o PEP for configurado para usar segurança COPS, a primeira mensagem *Client-Open* que o PEP envia ao PDP, deve ter um campo *Client-Type* preenchido com zero. Para além disso um objecto de integridade deve ser incluído, um objecto que contém um número sequencial de 32 bits. O PEP define o valor inicial do número sequencial que o PDP deve incrementar assim que o par de mensagens *Client-Open/Client-Accept* esteja concluído. Este sistema é usado para evitar ataques através de mensagens de resposta. Tanto o PEP como o PDP esperam receber uma mensagem com um determinado numero sequencial. Se o número sequencial de uma mensagem que chega é diferente do esperado o receptor envia uma mensagem de erro e fecha a ligação. Existem ainda dois outros campos no *Integrity –object*: o *KeyID* campo de 32 bits e o *Keyed message digest* campo de 96 bits. O campo *Key ID* é usado para identificar uma chave partilhada entre o PEP e o PDP bem como o algoritmo criptográfico a ser usado. O *check-sum* é calculado usando todos os campos dos objectos da mensagem, coma excepção do campo *Keyed Message Digest*. É esta a razão pela qual o objecto de integridade deve ser sempre o último objecto da mensagem, quando é usado. As implementações COPS devem pelo menos fornecer meios de manualmente configurar as chaves e os parâmetros localmente.

3.4 MODELOS DE GESTÃO DE QDS

Existem vários modelos de gestão de QdS com vantagens e desvantagens diferentes. Uma divisão destes modelos pode ser feita em termos de que entidade (ou entidades) toma as decisões de gestão e de que entidade as põem em prática. Cada modelo de gestão tem defensores que esgrimem argumentos em defesa dos modelos que preferem, (por exemplo ver o grupo de discussão *WG-QDS* do site *http://archives.internet2.edu*). Seguidamente descrevem-se esses modelos e apresentam-se

as vantagens e desvantagens de cada um deles. Apresenta-se também o modelo de gestão de QdS usado neste trabalho.

3.4.1 Gestão distribuída

Num modelo de gestão distribuída são os routers quem toma as decisões de gestão da rede, como ilustrado na Figura 36. Neste caso não existe nenhuma entidade superior que tenha qualquer influência nas decisões tomadas.

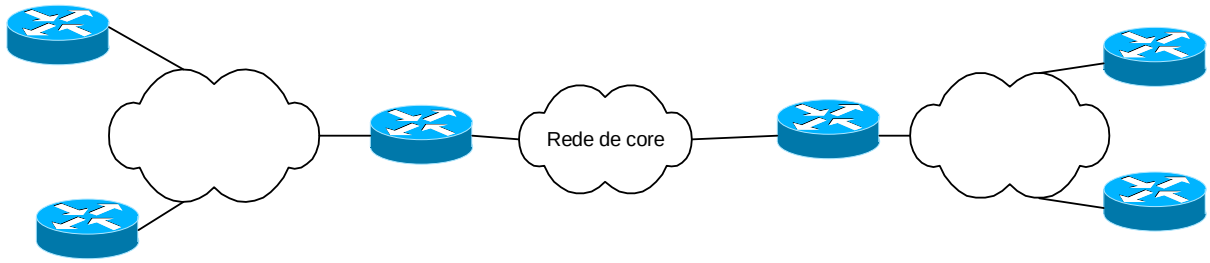

Figura 36 - Exemplo de gestão distribuída

Num modelo deste género pode-se ainda usar ou não controlo de admissão. Caso seja usado, este tipo de controlo assemelha-se ao modelo *IntServ* descrito no capítulo 2.1.1, onde através de RSVP os elementos de rede vão pedindo sucessivamente uns ao outros, autorização para encaminhar um fluxo com determinadas características. Esta solução tem graves problemas de escalabilidade que se manifestam à medida que o número de fluxos ou o numero de nós entre a origem e o destino aumentam, como aliás foi referido anteriormente quando se descrevia o *IntServ*. Caso o controlo de admissão não seja usado supõem-se que os recursos da rede de core são sempre abundantemente disponíveis e que quaisquer congestionamento se daria na rede de acesso. Apesar de frequentemente isto acontecer, um sistema deste género nunca pode dar garantias de QdS entre extremos.

Globalmente esta solução é mais simples e por conseguinte mais barata. Tende ainda a conseguir taxas de utilização dos recursos da rede mais elevadas do que as demais, mas com menor qualidade.

3.4.2 Gestão centralizada

Nos sistemas de gestão centralizada existe uma entidade superior que toma as decisões de gestão para todos os elementos da rede, a que tipicamente se chama um Bandwidth Broker e que se encontra esquematizado na Figura 37. O BB funciona como um PDP ao qual cada um dos elementos da rede faz pedidos antes de conduzir cada um dos fluxos do tráfego. As decisões tomadas pelo BB são aplicadas pelos elementos de rede que actuam como um PEP. O BB mantém em memória o estado de cada um dos routers podendo em qualquer altura que tenha que tomar uma decisão saber o estado de ocupação dos interfaces desse router. Todos os recursos são num modelo de gestão deste género controlados pelo BB, a que também se chama "oráculo".

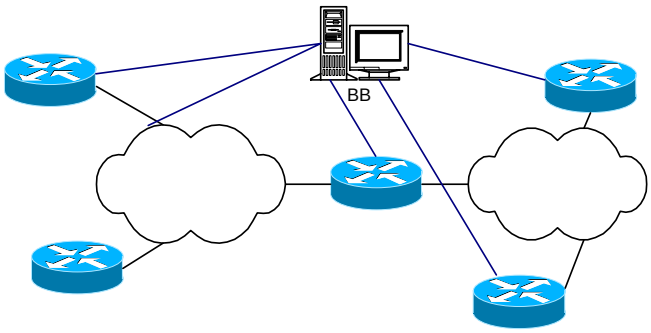

Figura 37 - Exemplo de gestão centralizada

Como características deste tipo de implementação podem se referir as seguintes:

- a elevada sobrecarga na rede provocada pela informação enviada na comunicação entre os routers e o BB;
- utilização dos recursos de rede pelos routers não ser tipicamente tão elevada como no caso de gestão distribuída. Este tipo de desvantagem pode no entanto ser anulada ou diminuída através de uma política de gestão mais inteligente implementada pelo BB;
- tem problemas de escalabilidade no que se refere ao número de routers que podem ser geridos por um só BB, devido à limitação da capacidade de processamento do BB.

Trata-se no entanto de uma solução bastante mais flexível do que a solução de gestão distribuída, podendo facilmente ser dotada de novas políticas de gestão sem que todos os elementos da rede precisem de ser mudados.

3.4.3 Gestão hierárquica

Um outro modelo possível, parcialmente explorado neste trabalho, é composto por dois níveis de gestão. Cada sistema autónomo possui um BB que gere todos os elementos pertencentes a essa mesma rede, e existe um Sistema de Gestão da Redes (NMS) que gere os recursos que cada BB tem disponíveis na rede de core. Esse tipo de arquitectura pode ser ilustrado pela Figura 38.

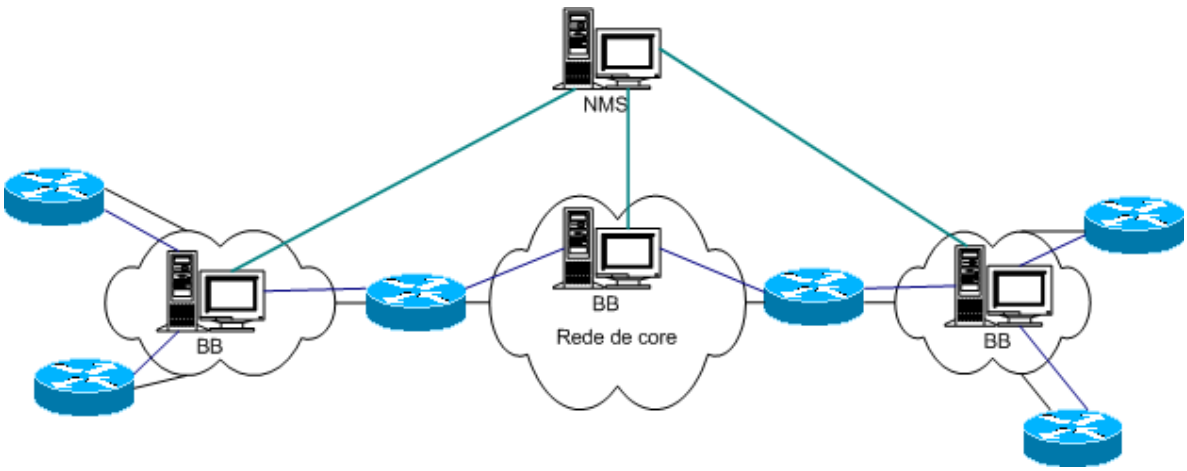

Figura 38 - Exemplo de gestão hierárquica

Cada BB de um sistema autónomo controla todos os elementos da sua rede funcionando do mesmo modo que foi descrito no capítulo anterior. Periodicamente são-lhe atribuídos pelo NMS de nível superior recursos da rede do core, os quais eles gerem e distribuem pelos pedidos da rede de acesso que recebem.

É de notar que o NMS gere aglomerados de tráfego e não fluxos em si. Como consequência disso, a frequência de oscilação da utilização dos fluxos é bastante mais baixa do que a frequência de aparecimento de um novo fluxo numa rede de acesso, o que leva a que a frequência de alterações dos recursos da rede do core disponíveis para cada sistema autónomo seja bastante mais baixa.

Como desvantagens desta implementação podem-se enumerar as seguintes:

- complexidade de implementação muito maior;
- sobrecarga provocada pelas mensagens entre BB e entre BB e routers é ainda maior;
- problemas de baixo aproveitamento dos recursos de rede é ainda maior do que no modelo anterior, e são mais difíceis de resolver devido à complexidade da arquitectura.

É no entanto de notar que este modelo tem algumas vantagens que o torna interessante:

- sistema muito mais escalável, podendo ser adicionado a um modelo destes novos sistemas autónomos sem que qualquer alteração na rede preexistente seja necessária, e sempre que um sistema autónomo crescer muito pode facilmente ser dividido em dois novos sistemas autónomos;
- sistema muito mais flexível, sendo muito simples adicionar servidores de políticas, ou servidores de AAAC;
- sistema facilmente estruturável por domínio administrativo.

Este modelo é bastante semelhante ao modelo usado no desenvolvimento do Bandwidth Broker que se descreve neste documento.

4. GESTORES DE QDS

4.1 O BANDWIDTH BROKER

Um Bandwidth Broker é a entidade que toma decisões de controlo de admissão e faz configuração dos componentes da rede, de modo a obter QdS entre extremos da rede, (eventualmente através de diferentes redes). Cada BB tem o seu próprio domínio de acção. Para se garantir QdS entre extremos, todos os BB's que estão ao logo do percurso se devem ligar entre si.

O BB gere e monitoriza os recursos da rede, tanto no seu domínio, como nas fronteiras. Também monitoriza as reservas de recursos de entrada e de saída nas fronteiras da rede. A informação recolhida é então usada em conjunto com a informação do Servidor de Políticas para tomar decisões de controlo de admissão. As SLS são implementadas pelo BB dentro de cada domínio, mas o BB também é responsável por gerir as comunicações entre domínios com os BB vizinhos, fazendo com que as SLS sejam coordenadas (negociadas) através de vários domínios. Esta coordenação pode ser feita agregando os fluxos de um determinado domínio que tem as mesmas características de serviço em termos de requisitos de QdS e tratar esses fluxos como um só. Dentro de cada domínio, um fluxo para ser admitido tem que enviar um pedido que vai ser analisado pelo BB em função do perfil do utilizador, bem como das disponibilidades dos recursos do sistema. Assim permite ao BB coordenar a alocação e aprovisionamento de recursos necessários para todos os fluxos e agregados, tanto dentro do domínio como para/de fora dele.

A Figura 39 [41] mostra um exemplo de uma configuração de rede simples. É constituída por três domínios AS1, AS2 e AS3 com BB's BB1, BB2 e BB3 respectivamente.

Os *Resource Allocation Request* (RAR) indicam pedidos de recursos ou serviços, de um utilizador individual ao BB do seu domínio. As respostas aos RAR designam-se de *Resource Alocation Answer* (RAA) e são enviadas pelo BB para o utilizador individual, após analisar o pedido recebido.

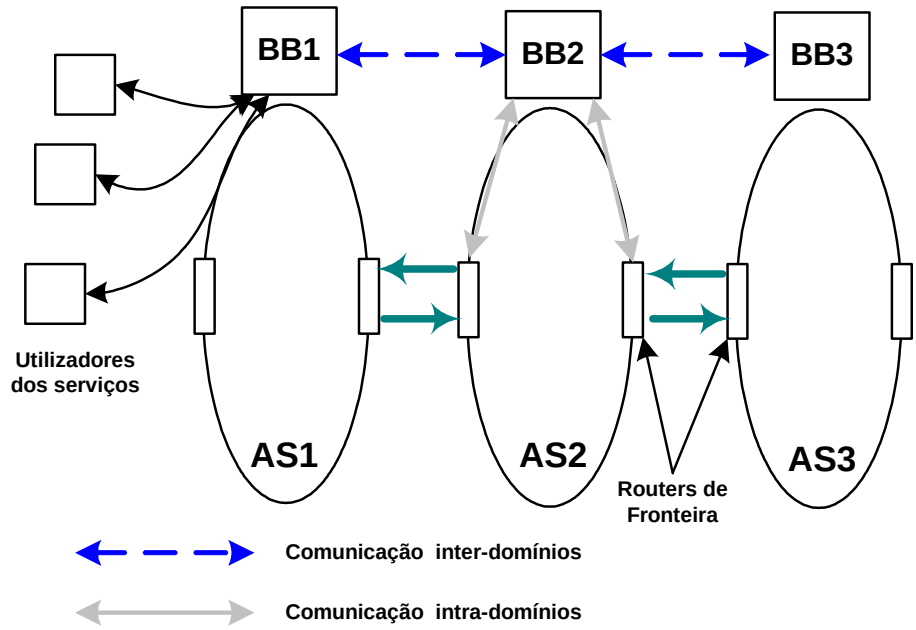

Figura 39 – Exemplo de uma rede simples

Da descrição das tarefas normalmente desempenhadas pelo BB, e considerando o estabelecimento de ligações ao longo de vários domínios, uma arquitectura típica para o BB pode ser ilustrada na Figura 40.

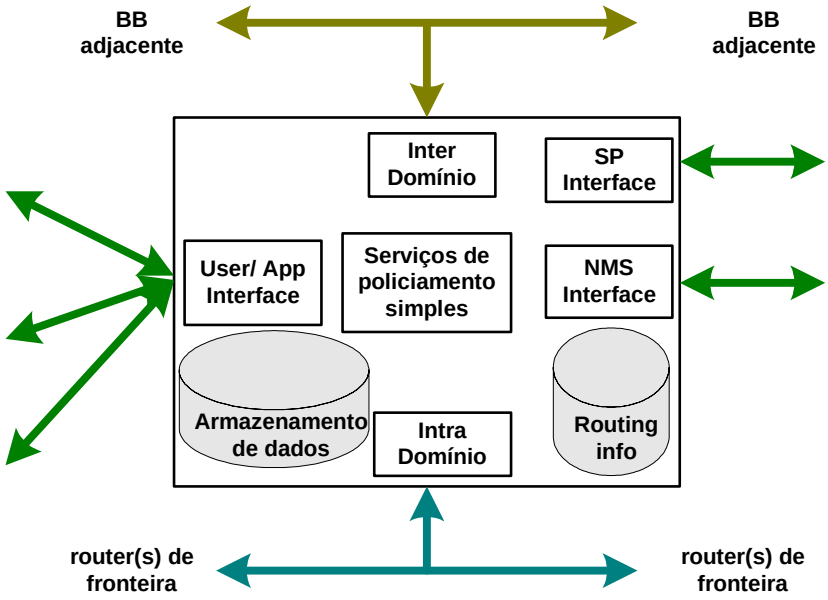

Figura 40 – Esquema interno de um BB

Esta arquitectura retira ao BB todas as tarefas de configuração da rede e planeamento de tráfego. Por exemplo, o núcleo da rede não é da competência do BB, mas de outra parte do NMS que deve monitorizar a performance global da rede e enviar comandos de reconfiguração do núcleo se for necessário. O BB terá apenas como responsabilidades gerir a configuração e optimização dos recursos da rede de acesso.

Os requisitos tradicionais de um BB são:

- Possuir interfaces com sistemas de gestão;
- Possuir interfaces com equipamentos de rede;
- Ter capacidade de controlo de elementos de rede;
- Ser uma aplicação escalável por forma a poder gerir os recursos de uma rede;
- Ser uma aplicação flexivel, podendo facilmente ser adptada a redes diferentes.

De modo geral, um BB tem cinco tipos de interface:

- com os BB vizinhos negociando informação de SLA entre os respectivos domínios;
- com os equipamentos de rede, enviando parâmetros de configuração dos router internos ao domínio de modo a garantir políticas de QdS, fazendo gestão de tráfego;
- com utilizadores/aplicações;
- com servidor de políticas
- com um NMS.

Internamente o BB tem quatro componentes principais:

- uma tabela eventualmente completa de roteamento do seu domínio;
- um repositório de dados que é utilizado por todos os componentes, e que pode eventualmente ser partilhado com o PS ou com NMS se existir;
- uma interface para o *Policy Manager* (PM) para coordenar os contractos de QdS e recursos de rede, para fornecer um controlo de admissões e funções de informação de AAA;
- interface de gestão para coordenação de aprovisionamento da rede e monitorização.

As implementações do BB podem variar muito com o tipo de rede. Por exemplo o equipamento de rede pode ser configurado através de SNMP [49], COPS (ver capítulo 3.3.2.2), ou *Lightweight Directory Access Protocol* (LPAD), originando esquemas de gestão muito diferentes. A escolha da arquitectura de servidor do BB influencia a complexidade do sistema; podem-se implementar um conjunto de serviços fixos, um conjunto parametrizável de serviços, ou uma descrição formal dos serviços (através de XML). Para além disso, a complexidade do BB é muito

dependente dos parâmetros de QdS suportados pela rede. Actualmente as implementações focam-se na limitação de Largura de Banda e em esquemas simples de prioridade.

4.2 IMPLEMENTAÇÕES EXISTENTES

Existem múltiplas implementações de Bandwidth Brokers, tipicamente correndo em sistemas operativos baseados em UNIX. A maior parte delas é desenvolvida em C, apesar de também existirem em Java [42]. Algumas das implementações fazem controlo de utilizadores como é o caso de [44], que inclui ainda informação dos clientes e das respectivas SLA numa base de dados.

Existem três implementações de Bandwidth Brokers muito conhecidas que serão brevemente discutidas nas secções seguintes.

4.2.1 KU-ITTC

A implementação do BB do *Centro de Tecnologia de Telecomunicações e Informação da Universidade do Kansas* (KU-ITTC) pode ser encontrada em [45]. Trata-se de uma implementação simples que permite somente gestão para dentro de um domínio em ambientes Linux, além de permitir interacção com routers Cisco.

4.2.1.1 Arquitectura

A arquitectura desta implementação é baseada numa arquitectura cliente-servidor típica, onde os terminais são os clientes que pedem reserva de largura de banda e o BB é um servidor que gere essa largura de banda e envia aos routers informação de configuração. A Figura 41 esquematiza a arquitectura e a interacção entre cada entidade no domínio.

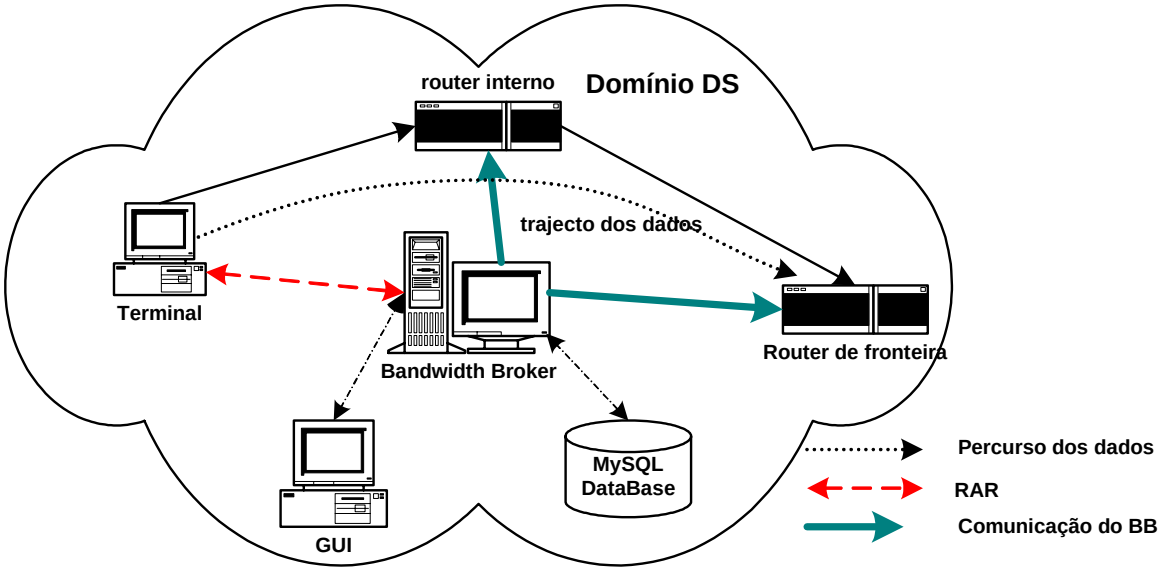

Figura 41 – Esquema da arquitectura da implementação do KU-ITTC

O servidor de BB desenvolvido pelo KU-ITTC corre numa máquina Linux, localizado no domínio, tipicamente um domínio *DiffServ*. A informação de políticas, as reservas de Largura de Banda, SLA's, entre outras, estão guardados numa base de dados. O servidor de Base de dados pode estar, ou não na mesma máquina. Para gerir as SLA's, e consultar outra informação relacionada com a utilização da Largura de Banda (como reservas) o sistema contém um *Graphical User Interface* (GUI) que pode ser acedido através de uma ligação http.

O servidor de BB interage com dois tipos de entidades: os terminais e os routers de fronteira. Os primeiros geram um pedido para especificar (ao servidor BB) a quantidade de Largura de Banda pretendida. Depois o BB responde ao terminal aceitando ou negando o fluxo, e podendo alterar a configuração do router se for necessário.

4.2.1.2 Operação

Antes de começar a funcionar o BB tem que ser configurado. Essa configuração envolve aspectos como os portos usados para resposta aos pedidos RAR, e definição de routers para gerir (endereço IP, tipo de router, etc.), etc.

A configuração das SLA's pode ser feita através do interface Web. Pode ser feita pelo Administrador quando recebe uma SLA enviada por um dos seus clientes, ou pode ser feita directamente pelos clientes que através do interface Web definem as suas SLA's. Esta implementação não suporta nenhum mecanismo de tarifação.

Quando um terminal quer enviar tráfego, comunica a largura de banda necessária ao Broker através de uma RAR. O Broker analisa a base de dados para ver se aquele cliente tem alguma SLA activa. Em caso afirmativo, analisa a largura de banda disponível e verifica se é possível atribuir a quantidade requisitada. Não sendo possível atender o pedido, o Broker responde com uma RAA negando a pretensão e assim rejeitando o fluxo. Se for possível aceitar o fluxo, um pacote com uma RAA positiva é enviado para o terminal, a base de dados é actualizada e o router pode ser reconfigurado.

A configuração dos routers é feita de modo diferente para os diferente routers que são usados:

- ***Routers Linux***. É aberto o *socket* TCP que comunica com o router *daemon* e actualiza a configuração.
- ***Routers Cisco***. É aberta uma sessão Telnet e são enviados comandos CLI.

Todas os pedidos de reserva (RAR) são limitados no tempo. Depois de um tempo definido as reservas expiram, do mesmo modo as reservas caducam sempre que as SLA's associadas são apagadas.

4.2.2 UCLA

A implementação do Bandwidth Broker de UCLA pode ser encontrada em [43]. O protótipo segue uma arquitectura *Two-Tier framework*[43] esquematizada na Figura 42, que assenta numa divisão entre o plano de gestão onde encontram os mecanismos para alocação de recursos para os vários fluxos, e o plano de reencaminhamento onde se incluem os mecanismos que criam a diferenciação de QdS para os pacotes reencaminhados

A Implementação faz somente gestão dentro do domínio e para ambiente (routers e terminais) FreeBSD.

4.2.2.1 Arquitectura

A arquitectura desta implementação não difere muito da arquitectura da implementação anterior. O BB continua a ser a peça central, mas nesta implementação só interage com os routers, não comunicando com os terminais. A Figura 42 esquematiza a arquitectura, bem como a interacção entre as diferentes entidades do domínio.

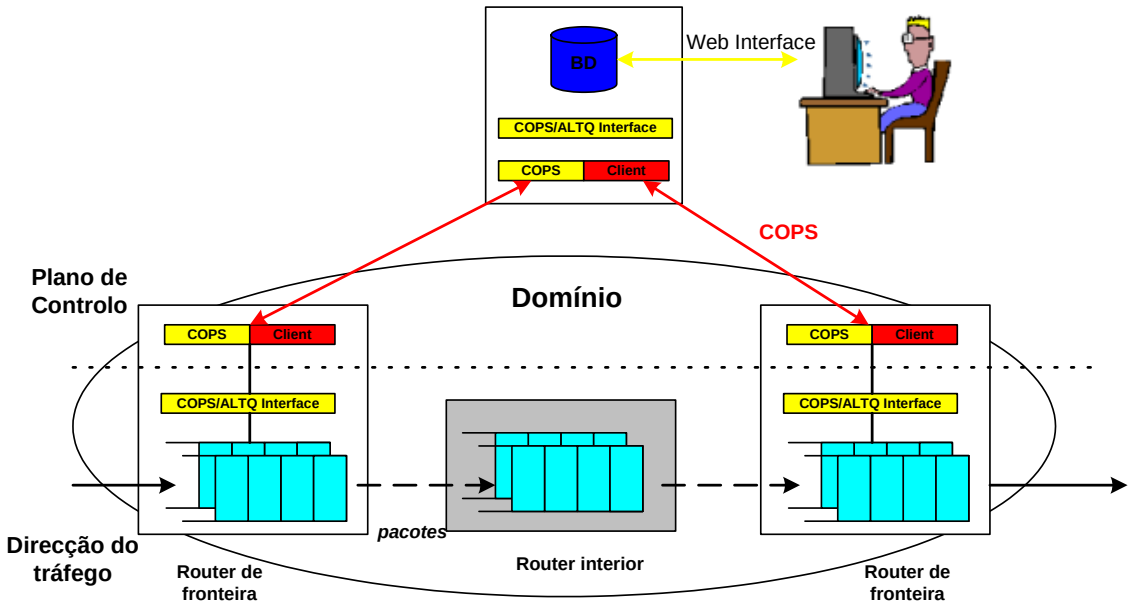

Figura 42 – Esquema da arquitectura da implementação da UCLAU-CSDIRL

O roteamento é feito da mesma maneira que é feito na implementação KU-ITTC, os pacotes saem do terminal e passam pelos routers que se encontram no caminho. O BB encontra-se numa máquina FreeBSD dentro do domínio. Usa uma base de dados para armazenar informação relacionada com as políticas do domínio, tais como reservas de tráfego, máxima alocação de largura de banda, time-out, etc. A base de dados tanto pode estar presente na mesma máquina, como pode estar num servidor de base de dados numa máquina á parte. A administração e consulta da informação de policiamento é feita pelos clientes e administradores da rede através de uma ligação http a um GUI-Web.

A grande diferença entre esta implementação e a anterior está na interacção entre o terminal, o router e o Broker. Neste caso não existe interacção entre o terminal e o Broker, os pedidos do terminal são transmitidos ao router de fronteira e as interacções são somente entre os routers e o Broker. Esta interacção é feita através do protocolo COPS.

4.2.2.2 Operação

É necessária a inicialização, tanto do BB, como dos routers de fronteira. O router precisa de informação do BB para saber que tratamento deve dar aos pacotes, que é lhe enviada través do protocolo COPS. Depois disso, quando um terminal envia tráfego, o router de fronteira trata-o de acordo com a política definida pelo BB. Pode acontecer que outros routers tenham que ser informados para que forneçam os recursos requisitados (e.g. routers de saída). Neste caso o router de entrada é responsável por o fazer. Quando um router de fronteira quer actualizar a informação

de política consulta o BB. Do mesmo modo se houver uma alteração das políticas o BB pode mandar essa informação para o router. Ambas as comunicações são feitas usando o protocolo COPS. O Broker actua como um PDP e o router de fronteira como um PEP.

São usadas três tipos de mensagens COPS: *request*, *decision*, e *report*. Existem dois cenários onde essas mensagens são trocadas:

- Quando o router pede ao Broker informação para se configurar. Neste caso envia uma mensagem *Request*, e o Broker responde com uma mensagem *Decision* que contém os parâmetros pedidos. Depois de ter recebido a mensagem, o router envia uma mensagem de *Report* mostrando o resultado da configuração;
- Quando o Broker envia uma mensagem para actualizar a configuração do router. Neste caso o Broker envia uma mensagem *Decision* para o router que responde com uma mensagem *Report* para comunicar o resultado da configuração.

4.2.3 QBONE

A arquitectura de implementação do QBone, a iniciativa norte-americana para construir uma plataforma de testes de novas tecnologias de QdS em IP, pode ser vista em [46]. Chama a esta visão do BB de "BB como um oráculo"[46], apesar de ser em tudo semelhante à arquitectura usada pela implementação da Universidade do Kansas.

4.2.3.1 Arquitectura

A decomposição funcional desta arquitectura pode ser representada pela Figura 40. Segue-se uma lista dos protocolos usados pelas interfaces mais importantes:

- ***Aplicação/Utilizador***. Esta interface é utilizada para fazer pedidos de alocação de recursos dentro do domínio do BB. Estes pedidos podem ser feitos manualmente (via interface Web) ou através de mensagens de RSVP;
- ***Dentro do domínio***. O objectivo da comunicação dentro do domínio é comunicar as decisões do BB aos routers do domínio, de modo a se transmitirem os parâmetros de configuração. Existem vários protocolos usados nas implementações existentes: COPS, DIAMETER, SNMP, e *Command Line Interface* (CLI);
- ***Entre domínios***. Neste caso o objectivo é a fornecer possibilidade de negociar com os Brokers adjacentes as decisões de troca de tráfego. O protocolo usado é o *Simple*

Interdomain Bandwidth Broker Signalling (SIBBS) [50], que se trata de um protocolo simples de pergunta-resposta entre Brokers adjacentes.

Existem duas bases de dados com modelos já definidos e com as interfaces implementadas:

1. ***Tabelas de encaminhamento***. O BB necessita de conhecer a informação de encaminhamento dentro do domínio para poder determinar o router de saída bem como os domínios de encaminhamento antes de poder aceitar uma RAA. Do mesmo modo o Broker precisa de conhecer a informação de encaminhamento dentro do domínio para poder determinar as rotas e assim determinar o estado de alocação dos recursos dentro do domínio.
2. ***Repositório de dados***. Este repositório de dados contém informação comum a todos os componentes do Broker. Pode ainda ser partilhada com componentes com que o Broker se relacione, por exemplo o controlador de políticas e gestor de rede. A informação presente nesta base de dados pode ser resumida na seguinte lista:
   - Informação de SLS para todos os routers de entrada e saída;
   - Reservas activas e atribuições de recursos;
   - Configuração dos routers;
   - Mapeamentos de serviços e mapeamentos DSCP;
   - Informação de Políticas;
   - Informação de gestão da rede;
   - Informação de monitorização dos routers;
   - Bases de dados com informação de autenticação e autorização tanto de utilizadores como de Brokers adjacentes.

De sublinhar que os BB adjacentes comunicam entre si através de SIBBS, negociando reservas de recursos, e assim criarem o designado “efeito de túnel”, que pode ser representado pela Figura 43, e que se discute na próxima secção.

4.2.3.2 Operação

O BB pode receber RAR’s tanto do exterior como do interior do domínio que controla. Responde com RAA, acedendo ou recusando o pedido feito. A resposta pode ter efeito na configuração dos routers, tanto nos routers de fronteira, como nos routers internos do domínio, para que disponibilizem os recursos apropriados.

O exemplo que se segue mostra o essencial da comunicação entre Bandwidth Brokers. A Figura 43 dá uma ideia da comunicação existente entre as entidades neste cenário. Os números existentes na figura e no texto mostram a ordem temporal das mensagens. É importante notar que as mensagens existem aos pares; cada pedido tem sempre uma resposta, seja ela afirmativa ou negativa.

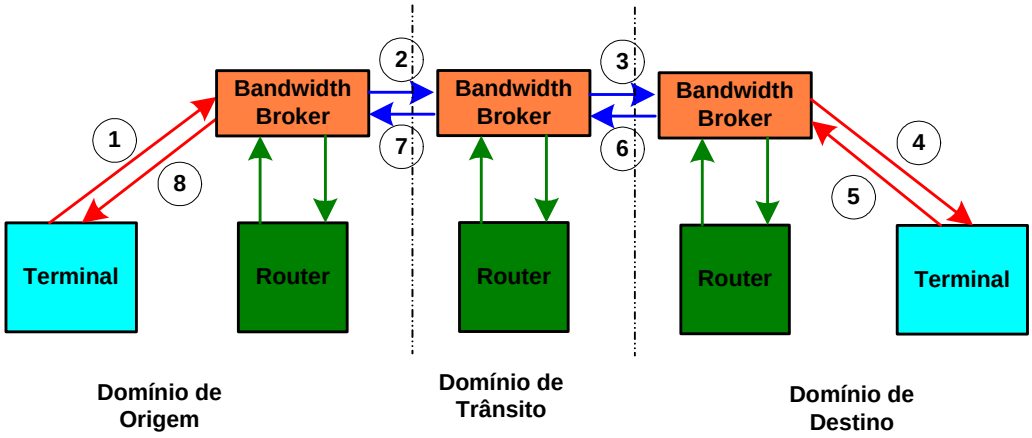

Figura 43 - Exemplo de comunicação entre Brokers

Inicialmente o sistema terminal envia uma RAR para o Bandwidth Broker (1). Esta mensagem inclui uma identificação do serviço, endereços destino e origem o campo de autenticação, os tempos nos quais o serviço é pedido outros parâmetros de serviço. Então o BB toma um conjunto de decisões:

- Se o sistema terminal está autorizado a ter aquele serviço;
- Qual o router de saída a que o fluxo pode ser assignado;
- Qual o caminho até ao router de saída;
- Se o fluxo está de acordo com o SLS que existe entre o router de saída e o domínio que aparece no caminho do fluxo;
- Quando o fluxo pode ser aceite, pelas regras das políticas do domínio, para um determinado serviço.

Se nenhuma destas questões fizer o Broker negar o pedido, vai colocar na RAR a identificação do domínio e entregar ao Broker que controla o domínio seguinte no caminho do pacote (2). Se o BB negasse o pedido respondia com uma RAA ao sistema terminal (8).

Quando o domínio de trânsito recebe a RAR de um BB adjacente, deve verificar as seguintes condições:

- Certificar-se de que o pedido é oriundo de um BB adjacente;
- Determinar o router de saída através das suas tabelas de encaminhamento;

- Certificar-se de que os recursos pedidos respeitam a SLS que tem com o domínio que lhe envia o pedido;
- Certificar-se de que os recursos pedidos respeitam a SLS que tem com o domínio a seguir no caminho do fluxo;
- Verificar se existem recursos para conduzir o fluxo desde a entrada do domínio até à saída e procurar o encaminhamento;
- Determinar se o fluxo respeita as políticas do domínio.

Se o fluxo passar todos estes testes o BB envia o pedido com o seu próprio ID para o domínio que se segue no caminho do fluxo (3). Caso contrário é enviado um RAA negando o pedido (7).

Quando o pedido chega ao domínio destino, o comportamento do Broker é o seguinte:

- Verifica que o pedido é enviado por um Broker vizinho;
- Determina o router interno que pode estabelecer a ligação desde o router de entrada até ao sistema final e decide que recursos estão disponíveis para aquele fluxo;
- Assegura-se que os recursos requisitados estão de acordo com a SLS do sistema final;
- Determina se o fluxo pode ser aceite, de acordo com as políticas do domínio em questão.

No caso de estas decisões terem sido no sentido de aceitar o fluxo o BB envia um RAR para o sistema terminal (4), com o objectivo de saber se este pode ou não aceitar o fluxo. A este pedido o siste,a terminal responde enviando um RAA para o broker do seu domínio (5). A RAA contém a identificação do sistema terminal, e os parâmetros fluxo que o terminal pode aceitar. No caso do fluxo ser rejeitado o RAA contém um código indicando a razão pela qual foi negado.

4.3 COMPARAÇÃO DE IMPLEMENTAÇÕES

A Tabela 3 sumaria uma comparação entre as implementações analisadas. As características comparadas foram o Sistema Operativo, o Sistema de Gestão de Bases de Dados (SGBD) utilizado, os protocolos de comunicação utilizados, o tipo de domínio, o meio de configuração do BB e o estado da investigação.

	KU-ITTC	***UCLA***	***QBONE***
Sistema Operativo	Linux	FreeBSD	Linux
Base de dados	MySQL	MySQL	MySQL
Comunicação com routers	ND	COPS	COPS, DIAMETER, SNMP e CLI
Comunicação entre domínios	N.D.	N.D.	SIBBS
Protocolo de sinalização	N.D.	N.D.	Mensagens RSVP
Tipo de domínio	*DiffServ*	*DiffServ*	*DiffServ*
Meio de configuração	Web GUI	Web GUI	N.D.
Estado de investigação	Parou	Parou em 1999	Continua
Interacção entre terminal e BB	Sim	Não	Sim

Tabela 3 - Comparação das diversas implementações

5. GESTOR DE QDS COM SUPORTE DE MOBILIDADE

5.1 REQUISITOS DA REDE DE TESTES

5.1.1 O projecto Moby Dick

O projecto Moby Dick, no qual este trabalho se insere, tinha dois objectivos principais: construir uma rede com tráfego proveniente de várias origens, disponibilizando a possibilidade de mobilidade entre redes de acesso e entre tecnologias; e integrar esse tráfego numa rede baseada em IPv6 mantendo mecanismos de QdS. A Figura 44 ilustra a arquitectura do projecto.

O projecto integrou tráfego proveniente de redes UMTS (WCDMA), WLAN (802.11), e Ethernet (802.3) numa mesma rede, dando assim resposta às tendências de integração de todo o tipo de infraestrutura através da tecnologia IP. Desenvolveu mecanismos que permitem a mobilidade entre redes de acesso e entre redes de tecnologias também diferentes, transportando o tráfego em IP versão 6, mantendo mecanismos de QdS. Para responder a esse objectivo, a rede Moby Dick funciona como uma rede *DiffServ* com vários serviços definidos com diferentes requisitos de QdS, em que a rede trata de maneira diferenciada os pacotes marcados como pertencendo a cada classe. Foram definidos serviços ao utilizador, baseados nos seguintes tipos conceptuais :

- ***Premium Service***: subscrição que permite ao utilizador utilizar a rede sempre que pretende, até um determinado limite;
- ***Regular Signature Service***: subscrição que define um serviço com parâmetros muito concretos (taxa de transmissão, atraso, perda de pacotes e disponibilidade);
- ***Serviço pré-pago***: subscrição que permite ao utilizador usar os recursos da rede até que um *plafond* previamente pago seja atingido.

O projecto *Moby Dick* inclui um servidor de AAAC para gerir toda a informação necessária à identificação de cada utilizador, do tipo de SLA que cada utilizador contractualizou com o seu operador de telecomunicações, e como extrair qual a SLS que define o comportamento da rede na presença de pacotes desse utilizador. O servidor faz ainda registo de utilização e tarifação.

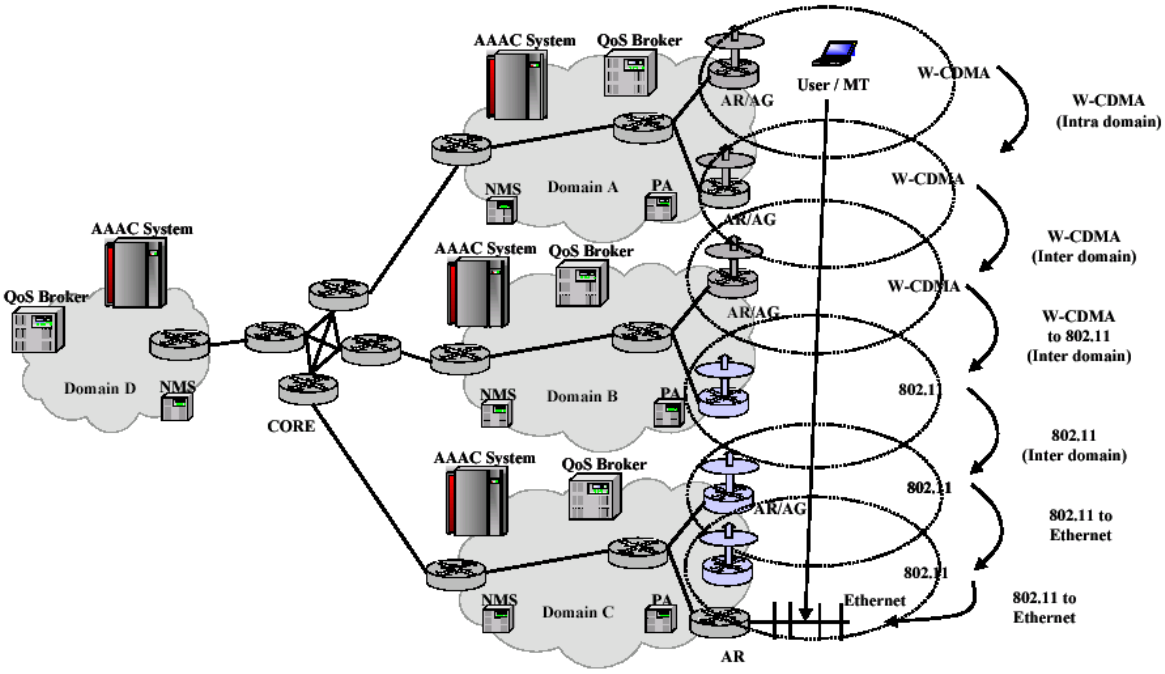

Figura 44 - Arquitectura da rede Moby Dick

A arquitectura do projecto *Moby Dick* diferencia o registo de um utilizador na rede da utilização de recursos da mesma. Quando um utilizador se autentica na rede *Moby Dick* o servidor de AAAC define-lhe o conjunto de serviços que este está autorizado a utilizar. O facto de um utilizador estar autenticado e autorizado a utilizar um serviço não significa no entanto que o possa fazer pois é necessário que existam recursos de rede suficientes para que este o faça. Quando um utilizador começa a utilizar os recursos de rede o router de acesso captura o primeiro pacote do fluxo e com o código DSCP, o endereço de origem e endereço de destino compõe um pedido de autorização de reencaminhamento do fluxo que envia ao BB. Após analisar os recursos disponíveis na rede o BB responde aceitando ou negando o pedido. Quando a resposta do BB é positiva o router de acesso conduz os pacotes do fluxo e é inicia uma sessão de contabilização de utilização dos recursos. A informação gerada nessa sessão é entregue pelo router de acesso ao AAAC.

5.1.1.1 Implementação de Mobilidade no projecto Moby Dick

O projecto *Moby Dick* utiliza ainda um processo de implementação de mobilidade designado de *Fast Mobile IP* [52] que permite que não exista perda de pacotes quando um terminal se move de uma rede de acesso para uma outra. O conceito assenta no facto de que a rede assim que detecta que um MT se está a mover de uma rede de acesso para uma outra inicia um processo de *bicasting* enviando os pacotes do MT por ambos os routers de acesso até que o processo de mobilidade seja concluído. Deste modo, assim que MT muda o seu ponto de ligação à rede, imediatamente lá

possui a informação que lhe é destinada. A Figura 45 ilustra o processo de mobilidade com as acções numeradas de acordo com a sua ordem temporal.

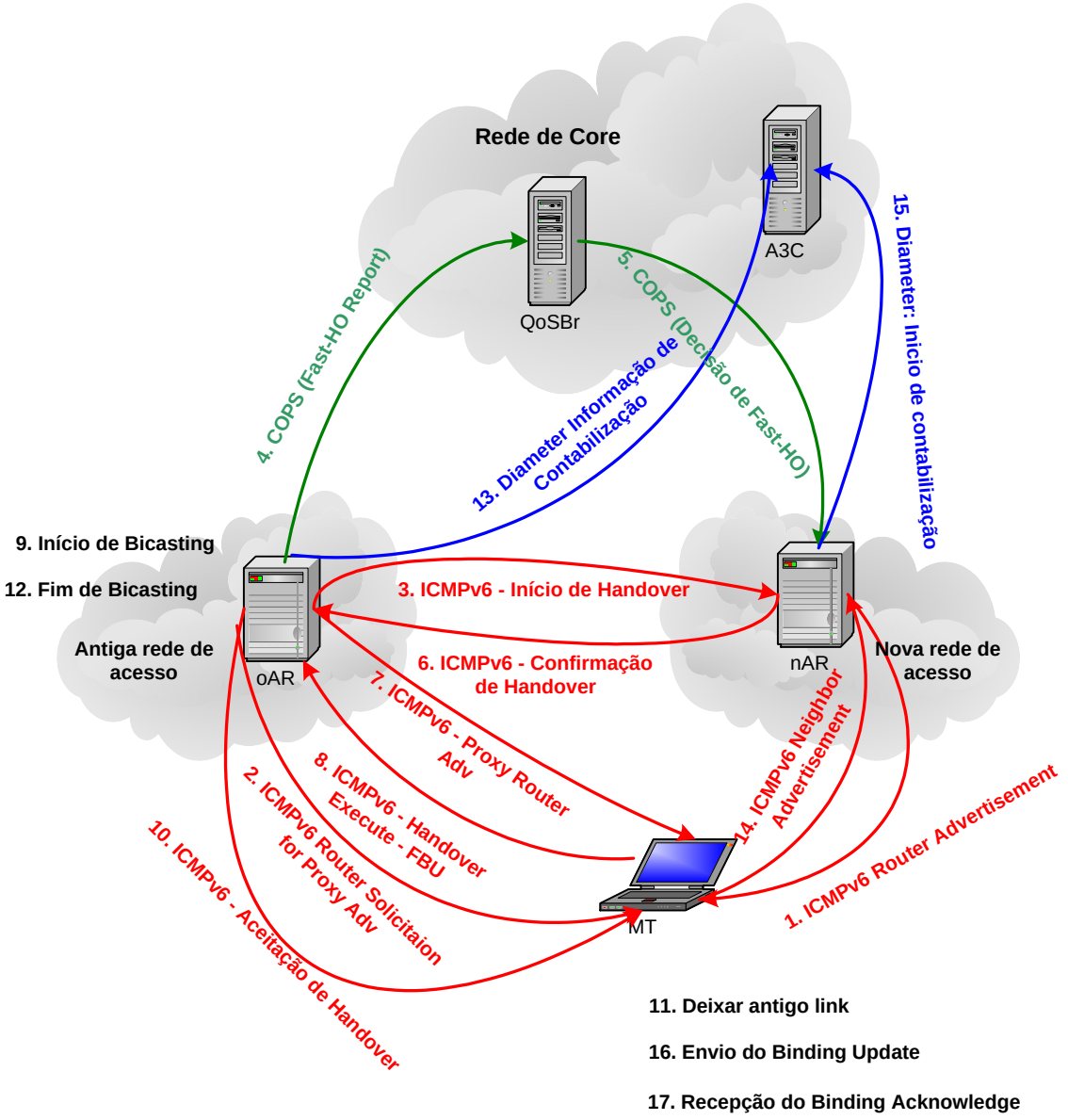

Figura 45 - Processo de handover na rede Moby Dick

Quando um terminal começa a perder sinal radio de uma célula, começa a receber sinal das células adjacentes (1). Este sinal identifica a célula e ao recebe-lo o terminal móvel (MT) inicia o processo de *handover* (2) através do seu router de acesso (oAR). O oAR contacta o router de acesso da nova célula (nAR) (3) para negociar com o MT todos os parâmetros (endereço em uso, etc). Depois o BB responsável pelo oAR é contactado por forma a autorizar o *handover* (4), enviando o antigo e o novo endereço de rede do terminal, bem como o endereço do nAR . No caso de o nAR pertencer a uma rede gerida por outro BB, o BB da rede antiga (oBB) envia a informação acerca

dos serviços em uso pelo MT bem como informação acerca do perfil do utilizador para o BB da nova rede (nBB). O nBB depois de verificar que existem recursos suficientes para aceitar o *handover*, envia uma mensagem configurando o nAR (5) para que ao utilizador sejam prestados os mesmos serviços que ele utilizava na antiga rede. O nAR contacta o oAR confirmando o *handover* (6) e este reencaminha essa informação para o MT (7). Ao receber essa informação o MT envia um pedido de execução de *handover* ao oAR (8). Ao receber o pedido o oAR inicia o processo de *bicasting* (9) duplicando para o nAR todos os pacotes que se destinam ao MT que se está a mover. Depois o oAR envia para o MT a mensagem de aceitação do *handover* (10), e o MT deixa a antiga ligação (11). O oAR termina então o processo de *bicasting* (12) deixando de duplicar para o nAR a informação de destinada ao MT e envia uma mensagem ao servidor de AAAC a anunciar o fim da sessão de contabilização (13). O MT envia uma mensagem a anunciar que faz parte da nova rede (14) e o nAR contacta o servidor de AAAC a anunciar o início da sessão de contabilização na sua rede (15). Depois o MT envia uma mensagem de *Binding Update* ao seu *home agent* notificando-o da sua nova localização (16), a que este responde com um *Binding Acknowledge* (17).

5.1.1.2 Componentes da rede

A rede Moby Dick é constituída por vários componentes, como se pode ver pelo diagrama Figura 46.

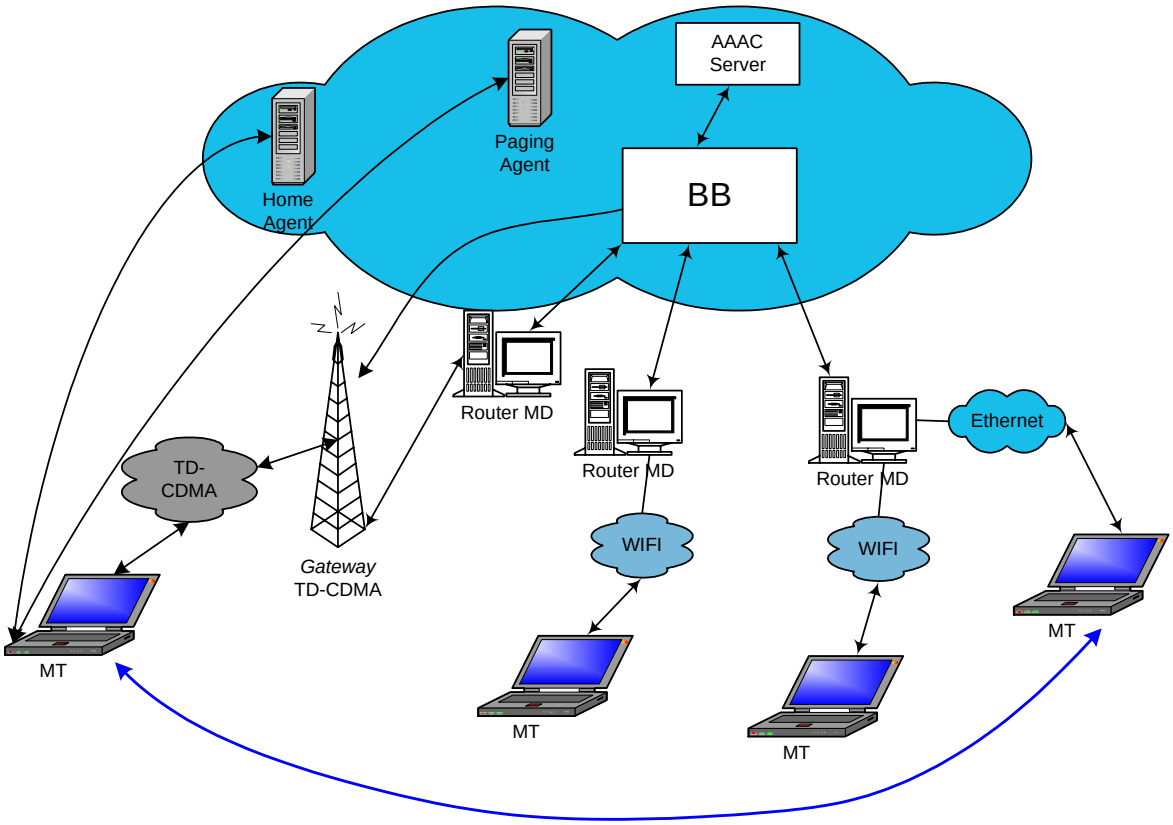

Figura 46 - Componentes da rede Moby Dick

Essas componentes podem ser enumerados na lista que se segue:

- ***Servidor de AAAC***. É a entidade responsável por autenticar a informação do utilizador, identificar a SLA do utilizador, e por manter a informação de utilização dos serviços e respectiva tarifação;
- ***Servidor do Sistema de Gestão de Rede (NMS)***. É a entidade que gere a configuração do núcleo da rede e que define em cada instante os recursos disponíveis para a rede de acesso gerida pelo BB;
- **BB**. Cada broker tem o seu domínio de acção. Para se poder fornecer mobilidade entre routers de diferentes Brokers, quando um broker recebe um pedido de handover para um router que não pertence à sua rede este envia os dados do pedido, a lista de serviços registados para esse utilizador, bem como a descrição do seu perfil ao broker que controla o novo router de acesso;
- ***Routers de acesso (AR)***. Os Routers tem que ser configurados de acordo com o perfil do utilizador e as condições de sobrecarga da rede, de modo a que se consiga garantir QdS. O QdS Broker é o responsável pela configuração dos routers;
- ***Home Agent (HA):*** agente que controla o processo de handover e que responde pelo MT durante o processo de mobilidade;

- ***Paging Agent(PA):*** agente que coordena o processo de paging onde o MT, em momentos em que não esteja a enviar ou receber tráfego, pode entrar em modo *dormant* podendo assim poupar bateria e recursos de rede. O PA é a entidade encarregue de o representar enquanto está no modo adormecido e despertar no caso de a sua resposta ser necessária;
- ***Mobile Terminal:*** terminal móvel que se desloca pela rede de acesso criando tráfego;
- ***Correspondent Node:*** terminal com o qual MT comunica;

5.1.2 Rede de Testes do Instituto de Telecomunicações

A plataforma de testes do Instituto de Telecomunicações, ilustrada pela Figura 47, possui duas grandes diferenças em relação à rede de testes do projecto Moby Dick:

- Não possui TD-CDMA, e por isso neste caso o BB não necessita de controlar a RG;
- Inclui routers Cisco, o que obriga a que o BB os possa controlar de uma forma diferente;

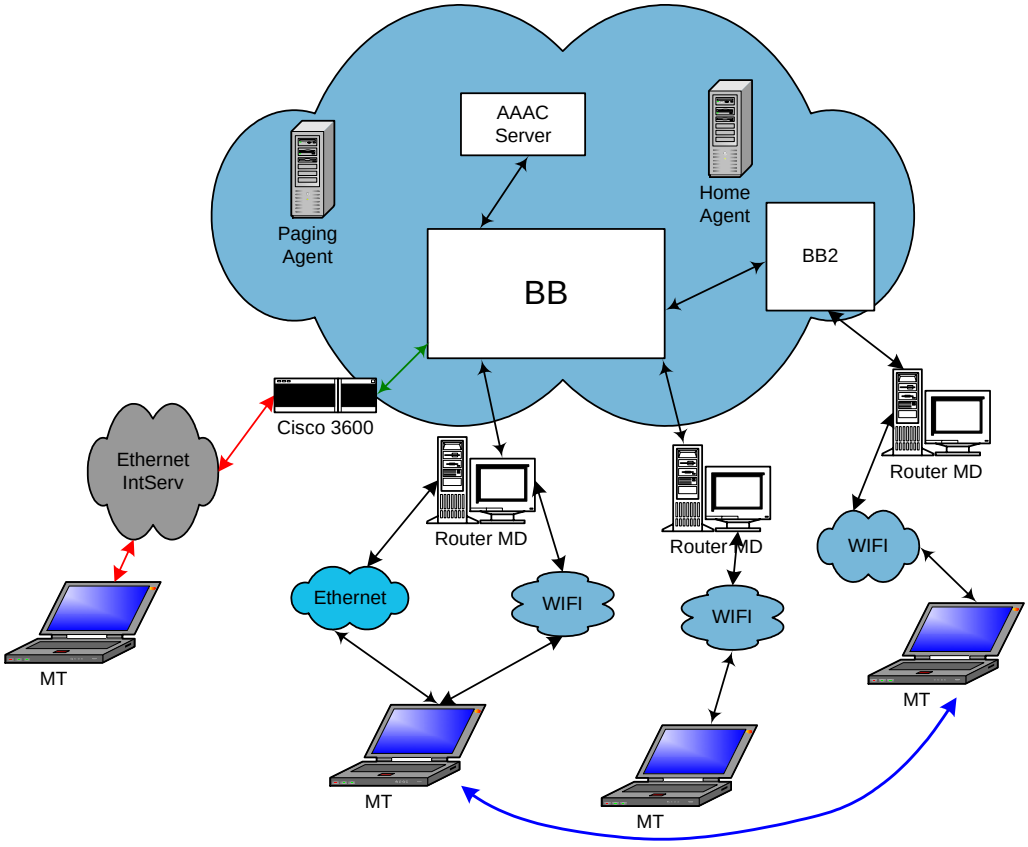

Figura 47 - Rede do Instituto de Telecomunicações

Para poder cumprir o objectivo de integrar equipamento de rede da Cisco, foi necessário desenvolver suporte para o mesmo. A integração foi feita utilizando COPS-RSVP configurando os routers Cisco como PEP's do BB. Segundo a implementação da Cisco do COPS-RSVP, o envio da informação dos pedidos *IntServ* é feito encapsulando o pacote RSVP do pedido *IntServ* em objectos do tipo *Client Specific Info* dentro do pacote REQ do COPS.

Devido às limitações de interacção entre o PEP e PDP do modelo COPS-RSVP foi ainda necessário desenvolver suporte CLI por forma a que outras acções de configuração pudessem ser executadas. A descrição da arquitectura deste interface do BB é descrita nas secções que se seguem.

5.2 ARQUITECTURA DO BB

Analisando as características da rede e tendo inferido os requisitos de funcionamento do BB definiu-se uma arquitectura, representada no diagrama de blocos da Figura 52 .

Pode-se observar as interfaces com os vários elementos da rede em como os principais elementos internos que constituem o BB. Segue-se agora uma lista dos componentes com uma breve descrição.

5.2.1 Características gerais do software

As características gerais do software desenvolvido podem ser enumeradas na lista que se segue:

- ***Código portável***: desenvolvido em ANSI C++, e inclui somente librarias de *POSIX threads e mysql,* podendo ser portado para outras plataformas uma vez que ambas as librarias não estão limitadas à plataforma usada;
- ***Código modular***: o programa foi desenvolvido de uma forma modular sendo muito fácil adicionar-lhe suporte para um novo elemento da rede;
- ***Suporte IPv4 e IPv6***: o programa foi inteiramente desenvolvido com suporte para ambas as versões de IP, permitindo que seja utilizado a gerir elementos de rede IPv4 ou IPv6;
- ***Baseado em threads***: o software foi desenvolvido de modo a ser um servidor escalável tendo por isso threads independentes que lhe permite simultaneamente manter diálogo com vários AR's e com os outros elementos de rede;

- ***Desenvolvido para Linux***: programa desenvolvido para plataforma Linux, foi testado com sucesso em várias distribuições de Linux da RedHat, SuSE e Mandrake;
- ***Programa robusto***: programa é bastante robusto, especialmente na sua ultima versão, graças ao longo período de testes e aperfeiçoamentos a que esteve submetido durante a fase final do projecto.

5.2.2 QBrokerEngine

QBrokerEngine é o módulo central do BB. É responsável por gerir as diferentes autorizações, por fazer controlo de admissão dos pedidos de rede. É ainda responsável por gerir um conjunto de *threads* que fazem parte do BB. Essas *threads* escutam pedidos dos componentes de rede exteriores ao BB. São elas:

- NMSAttendant –Thread que atende os pedidos vindos do interface NMSInterf;
- A3CAttendant – Recebe e responde aos pedidos vindos do servidor de AAAC.
- InterBrokerAttendant – InterBrokerAttendant vai responder aos pedidos provenientes de outro Broker. As mensagens que sinalizam os processos de handover provenientes de um AR de outro BB são recebidas por este interface;
- RouterAttendant – RouterAttendant vai receber pedidos de admissão de clientes por parte dos routers da rede.

5.2.2.1 Arquitectura do QBrokerEngine

A Figura 48 mostra um pormenor de QBrokerEngine, com os seus componentes internos e com os seus interfaces para o exterior.

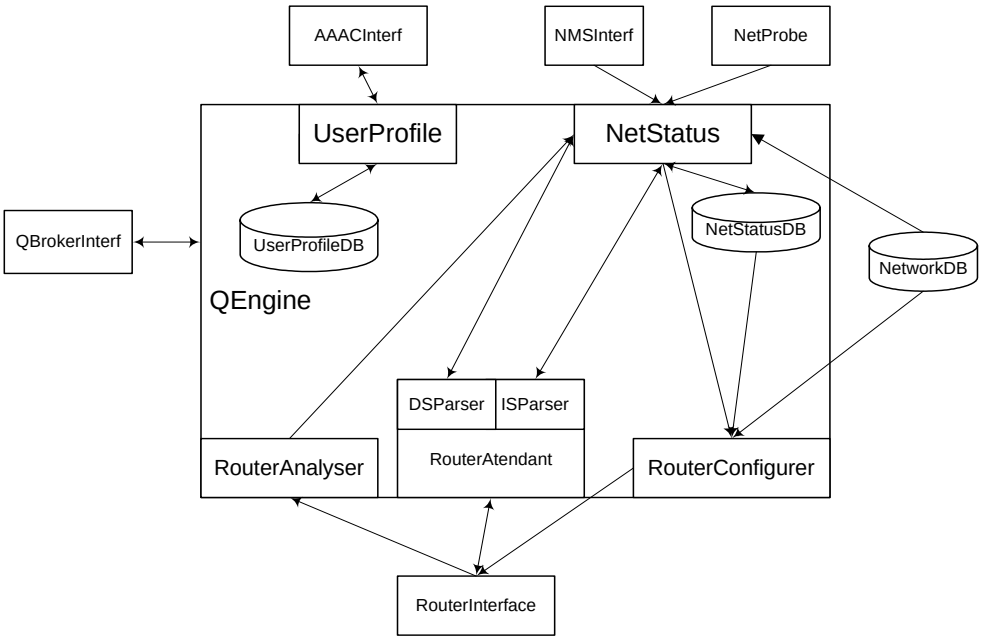

Figura 48 - Pormenor de QBrokerEngine

QBrokerEngine pode ser dividido nos seguintes vários módulos:

- ***UserProfile***: elemento as autorizações e perfis dados pelo sistema de AAAC. Utiliza uma base de dados chamada *UserProfileDB* onde mantém a informação acerca dos perfis de qualidade de serviço e dos perfis dos utilizadores autorizados pelo sistema de AAAC. É consultada sempre durante o processo de decisão de atribuição de um recurso, de modo a poder definir se um pedido está dentro das condições definidas no perfil do utilizador;
- ***NetStatus***: elemento que controla o estado da rede. Utiliza uma base de dados chamada NetworkDB onde lê informação acerca dos elementos de rede que pertencem ao domínio;
- ***RouterConfigurer***: é utilizado na configuração dos routers. Sempre que na sequência de um pedido ou na caducação de um recurso for necessária a reconfiguração de um elemento de rede, o módulo *RouterConfigurer* é chamado a fazer essa configuração. Utiliza a informação da base de dados *NetWorkDB* para ver qual a sequência de comandos que deve usar para cada acção de configuração que pretenda utilizar num determinado router;
- ***RouterAnalyser***: tem como objectivo ler do router informação acerca da tabela de encaminhamento, e do estado de sobrecarga das suas filas. Utiliza o router interface para conseguir comunicar com o router. Contem uma *thread* que está a correr e que de tempo a tempo lê as informações.

- ***RouterAtendant***: é o módulo que recebe os pedidos de recursos por parte dos routers. Tem um interpretador de pedidos *IntServ*, e um de pedidos dos routers Moby Dick.

5.2.2.1.1 UserProfile

O *UserProfile* é o módulo do BB que gere a informação dos utilizadores que estão registados na rede que o BB gere. Entre as tarefas executadas por este módulo podem-se listar:

- manter uma lista com os parâmetros de QdS definidos pelo servidor de AAAC para cada serviço disponível na rede;
- manter a lista de utilizadores registados na rede bem como a lista de serviços que cada um dos utilizadores está autorizado a utilizar;
- verificar se cada serviço pedido pelo AR para um utilizador está autorizado no perfil do mesmo;
- gerir tempo de validade de cada perfil de modo a que as autorizações caducadas sejam removidas da lista.

O *UserProfile* possui uma *thread* que está continuamente a correr e que periodicamente verifica cada um dos perfis de utilizadores registados e cada um dos serviços existentes. Para isso o servidor de AAAC renova periodicamente os perfis e os serviços definindo o tempo de validade de cada serviço.

5.2.2.1.2 NetStatus

O NetStatus é o módulo do QBrokerEngine que controla o estado dos vários elementos da rede que o BB gere. Tem ligações aos seguintes elementos:

- ***NMSInterf***: interface de NMS responsável pela configuração de toda a rede e do próprio BB;
- ***NetProbe***: receberá por este módulo informação do estado da rede que adicionará à informação recolhida pelo módulo *RouterAnalyser* e assim poder ter uma melhor noção do estado da rede;
- ***RouterConfigurer***: é utilizado pelo *NetStatus* para promover qualquer configuração necessária em qualquer router;
- ***RouterAnalyser***: é utilizado pelo NetStatus para periodicamente ler dos routers informação acerca das suas tabelas de encaminhamento e do seu estado de sobrecarga.

Essa informação em conjunto com a informação recolhida pelo *NetProbe* é inserida na base de dados *NetStatus.*

No inicio da sua operação o NetStatus recolhe informação. Copia uma parte dessa informação sempre que o BB é inicializado para uma base de dados chamada *NetStatusDB.* Essa base de dados é mantida em memória e contem os dados mais frequentemente acedidos acerca dos elementos de rede. A Figura 49 ilustra a composição do módulo *NetStatus*.

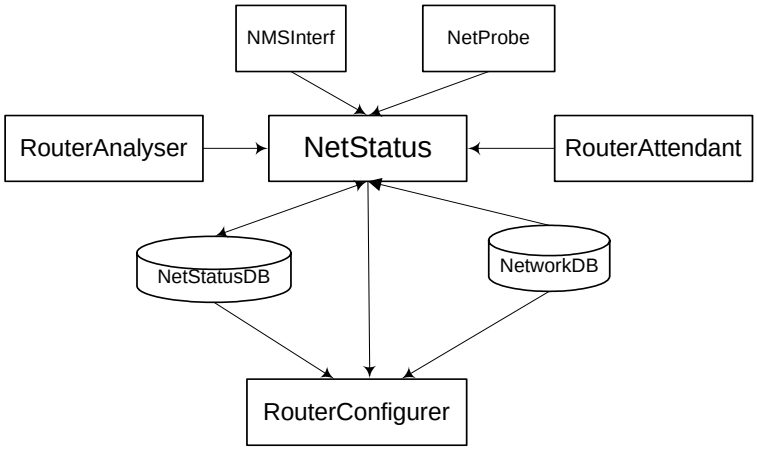

Figura 49 - Composição do NetStatus

5.2.2.2 Definição do algoritmo de controlo de admissão

Sempre que o BB recebe um pedido de recursos, precisa de tomar uma decisão. Para isso precisa de saber quais são os recursos que o pedido requer, e como está o estado de sobrecarga da rede nesse instante. O algoritmo de decisão utilizado pode ser representado pelo diagrama de fluxos representado na Figura 50.

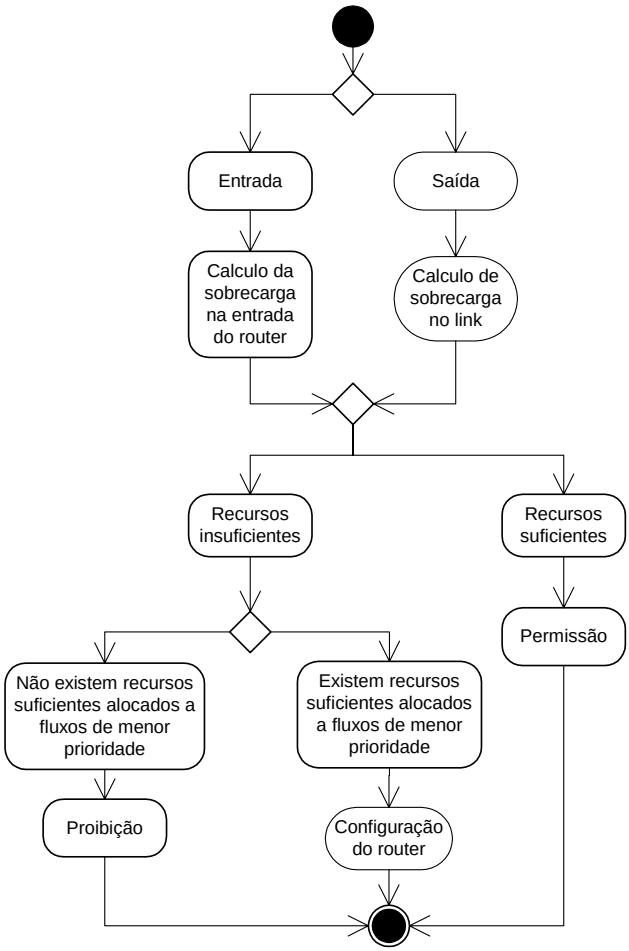

Figura 50 - Diagrama de fluxo do processo de decisão

O processo de decisão da parte do Broker pode então ser definido nos seguintes passos:

- classifica fluxo como sendo de entrada ou de saída;
- para um fluxo de saída calcula a sobrecarga do link de saída;
- para um fluxo de entrada calcula a sobrecarga de um link de entrada;
- Analisa se os respectivos fluxos são suficientes, segundo regras que entram em consideração com tipo de serviço e ganhos de multiplexagem;
- Se forem permite o acesso;
- Se não forem:
    - Testa se existem recursos suficientes alocados a fluxos de menor prioridade;
    - Se houverem:
        - aceita o fluxo;
        - configura o router de modo a remover os fluxos de menor prioridade;
    - se não houverem rejeita o fluxo;
- O processo é terminado.

5.2.3 NetProbe

O *NetProbe* é a entidade que monitoriza o estado da rede. A informação que vai sendo recolhida é inserida numa base de dados chamada NetStatusDB.

A base de dados será colocada em memória devido aos motivos que a seguir se referem:

- a informação a aí colocar será extremamente volátil: a informação que se irá manter valida durante mais tempo será a da tabela QoSProfile que durará enquanto o BB correr. Toda a outra informação tem um prazo de validade de alguns segundos. Apesar de uma das tabelas durar toda a sessão do BB essa tabela ocupa somente umas desenhas de bytes não constituindo por isso nenhum tipo de problema o espaço de memória por si ocupada;
- a quantidade de informação destas tabelas é relativamente pequena alguns milhares de bytes, não constituindo grande problema para a máquina mante-la em memória;
- a frequência de consulta desta informação é enorme sendo bastante importante a rapidez de consulta desta informação para o desempenho global do BB;

Optou-se assim por criar uma classe que representa esta base de dados e utilizar classes de STL (Standard Template Library) para alocar e gerir o array de estruturas que constituem as tabelas. Esta informação em conjunto com a informação recolhida pelo *NMSInterface* representa o estado da rede em cada instante. O *QBrokerEngine* vai usar esta informação para decidir se deve ou não permitir o acesso a um recurso de rede.

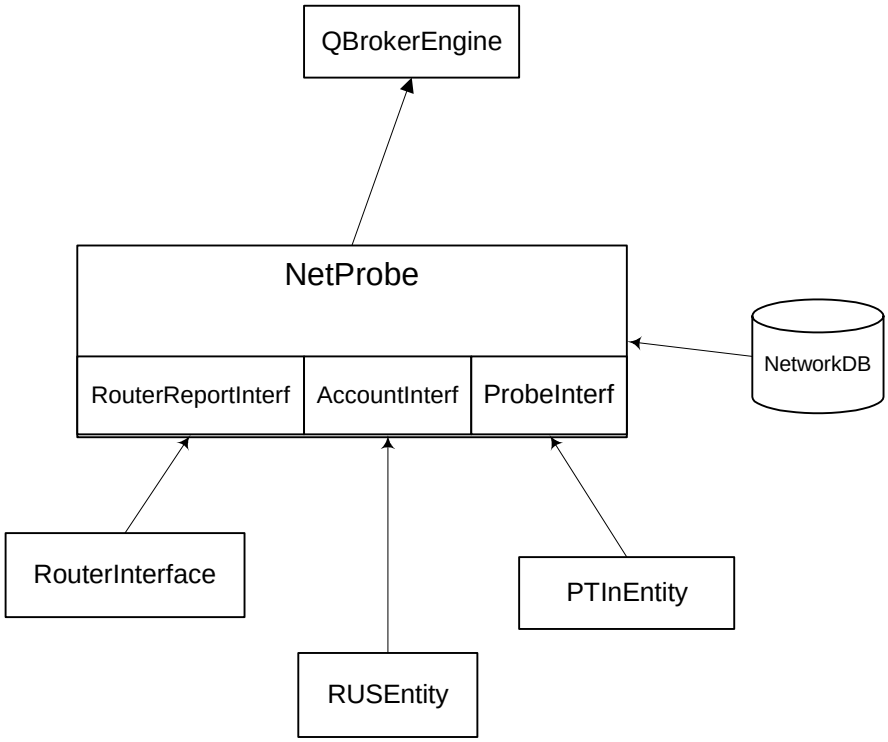

Figura 51 – NetProbe

Como se pode ver na Figura 51 são parte do NetProbe:

- ***RouterReportInterf***: módulo que processa e entrega ao QBrokerEngine os dados que são reportados pelos routers relativos ao tráfego que tem passado pelas suas filas;
- ***AccountingInterf***: módulo responsável por comunicar com entidade *RUSEntity*, desenvolvida pela Universidade de Estugarda. Vai também processar e enviar esses dados ao *QBrokerEngine*;
- ***ProbeInterf***: módulo responsável por comunicar com o módulo *PTInEntity*, módulo desenvolvido pela *PT Inovação*, e que se vai ocupar da monitorização do estado da rede. O *ProbeInterf* vai ainda tratar esses dados e entrega-los ao QBrokerEngine;

5.2.4 Interfaces do BB

A Figura 52 mostra um esquema do BB, em conjunto com os outros módulos pertencentes ao projecto Moby Dick onde se podem ver os interfaces.

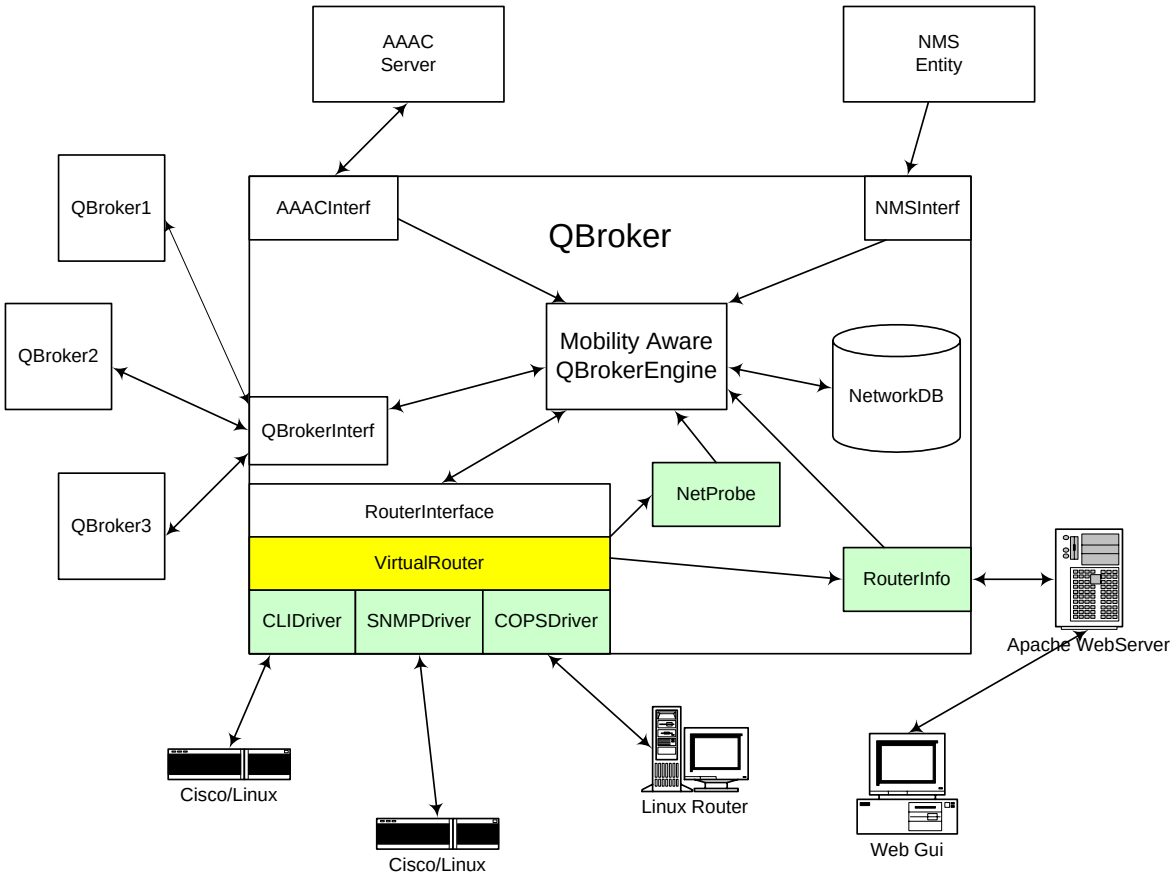

Figura 52 - Interfaces do BB na rede Moby Dick

O texto que se segue enumera as interfaces existentes no BB, mostra as mensagens que são enviadas por cada uma delas, e define os parâmetros que constituem cada uma dessas mensagens.

5.2.4.1 Interface com sistema de AAAC (AAACInterface)

O AAACInterface define o interface entre o BB e o sistema de AAAC. É através deste interface que o broker vai receber informação acerca dos serviços fornecidos pelo domínio do AAAC, dos utilizadores que estão autenticados e ligados ao domínio do BB. Recebe ainda por este interface a informação acerca do perfil de cada utilizador, designado pelos parceiros do projecto por *Network View of User Profile* (NVUP). Esta parte do perfil do utilizador contem o endereço em uso pelo terminal e a lista de serviços que o utilizador está autorizado a utilizar.

5.2.4.1.1 Mensagens trocadas

Existem várias mensagens que devem ser trocadas entre o AAAC e o BB:

- ***Antes do funcionamento da rede, altura em que a rede inicia funcionamento, que o BB arranca, ou que os serviços da rede mudam***: ServicesDescription – Esta mensagem é enviada pelo sistema de AAAC de modo a informar o BB dos serviços fornecidos no domínio do operador. Esta mensagem identifica os serviços suportados, com as suas características (largura de banda e prioridade) e os códigos DSCP de referencia associados. Esta mensagem pode ser vista como uma mensagem de configuração, anterior à entrada em operação do BB.
- ***Registo:*** *AuthorizeProfile* – O sistema de AAAC informa o BB da presença de um utilizador já autenticado no seu domínio, enviando-lhe uma parte dessa informação de autenticação (NVUP). Isto permite ao BB saber como processar os pedidos de recursos desse utilizador. Os recursos não estão reservados ainda, só são reservados no início da sessão.
- ***Anulação de Registo:*** *DeAuthorizeProfile* – O sistema de AAAC informa o BB para cancelar o perfil de um utilizador.
- **Durante o funcionamento:**
    - *RequestNVUPValidation* – Mensagem usada pelo BB pedindo ao sistema de AAAC a validação da informação de NVUP de um determinado utilizador, utilizando o handle para identificar o NVUP a ser validado;
    - *ValidateNVUP* – Mensagem enviada pelo sistema de AAAC validando a informação de NVUP utilizando o handle que identifica esse perfil. Este género de mensagem pode ser enviada em situação que envolvam serviços pré-pagos em que se exceda o limite de utilização.

5.2.4.1.2 Formato dos dados trocados entre o BB e AAAC

Segue-se a lista das mensagens que são trocadas entre o BB e o servidor de AAAC bem como a sua estrutura de dados:

- ***ServiceDescription*** – define os serviços de rede disponíveis no domínio do AAAC.

Os parâmetros trocados por cada serviço de rede são:

- ***DSCP code:*** código DSCP que identifica as condições de QdS do serviço;
- ***Bandwidth:*** largura de banda do serviço***;***
- ***Priority:*** prioridade dos pacotes do serviço***;***

- ***Delay:*** atraso dos pacotes;
- ***Destination Address:*** endereço destino dos pacotes.

O código DSCP funciona como a chave primária da tabela que descreve os serviços de rede, ServiceDescription.

NVUP (Network View of User Profile) - Representa um subconjunto da informação que descreve o perfil do utilizador que os elementos da rede necessitam conhecer. Este subconjunto inclui:

- ***UserID:*** identificação do utilizador;
- ***CareOfAddress:*** corresponde ao endereço de origem em cada momento;
- ***NServices:*** número de serviços que em cada momento o utilizador está autorizado a utilizar;
- Lista de N elementos com ***NetService*** - que descreve um serviço de rede em termos de:
  - ***endereço destino:*** será usado somente usado em casos especiais tais como número de emergência;
  - ***Validade da autorização:*** tempo de validade do serviço;
  - ***código DSCP:*** valor que define os parâmetros de QdS para o serviço de rede em questão. O BB vai então procurar na tabela ServiceDescription quais os parâmetros de qualidade de serviço que são apontados por este valor.

5.2.4.2 Interface BB – Router de acesso (RouterInterface)

Esta interface é baseada no protocolo COPS. Existe uma API (COPS-ASM) que foi integrada no RouterInterface de modo a poder formatar e ler as mensagens em COPS. Esta interface entrega ao QBrokerEngine os pedidos que recebe dos AR, e espera pela resposta do QBrokerEngine que devolve ao AR.

5.2.4.2.1 Mensagens trocadas

- ***RequestResource***: mensagem enviada pelo router pedindo ao BB recursos definidos por um DSCP para um utilizador;

- ***ReleaseResource***: mensagem enviada pelo router libertando os recursos previamente alocados para um utilizador;
- ***ConfigRequest:***: pedido do router a pedir ao BB para enviar a informação de configuração como a definição das filas e classes de serviço, dos filtros, etc. Esta mensagem é enviada na altura em que o router arranca;
- ***ChangeConfig***: envio de nova configuração do BB para o router;
- ***HandoverRequest***:: pedido enviado pelo oAR a solicitar ao BB que permita o processo de handover do MT para um outro AR.

5.2.4.2.2 Formato dos dados trocados entre o router e BB

- ***RouterConfig***: informação de configuração que envida pelo BB para o router. Esta informação compreende:
    - ***DSCP:*** código DSCP da classes de serviço associado a cada fila***;***
    - ***BW:*** largura de banda do fluxo de pacotes reservado à aquela fila***;***
    - ***Borrow_flag:*** *flag* indicativa se fila deve ou não ceder tempo de serviço a outras filas***;***
- ***RequestResource:*** descreve um recurso que é pedido por um utilizador. O router de acesso envia a seguinte informação ao BB:
    - ***Endereço origem:*** endereço origem do fluxo;
    - ***Endereço destino:*** endereço destino do fluxo;
    - ***Código DSCP de sinalização:*** código DSCP que define a classe de serviço do fluxo.

O código DSCP identifica os parâmetros de qualidade de serviço definidos na tabela de *ServiceDescription*. Com esta informação o BB vai analisar as condições de sobrecarga da rede e aceitar ou rejeitar o pedido.

- ***HandoverResource***: descreve quais os recursos que o MT após o processo de handover pretende utilizar. É constituída pelos seguintes dados:
    - ***oCoA:*** antigo CoA do MT;
    - ***nCoA:*** novo CoA do MT;
    - **nAR:** endereço do novo AR para qual o MT pretende mudar.

5.2.4.3 Interface entre BB e NMS (NMSInterface)

A NMSInterface é a interface pela qual o BB vai conhecer os recursos disponíveis na rede de core. A definição desses recursos vai ser feita através de uma mensagem chamada SendQoSConfig. A estrutura dos dados é a seguinte:

QoSResourcesData

- ***UpstreamBW:*** largura de banda *upstream* disponível;
- ***DownstreamBW:*** largura de banda disponível *downstream*;
- ***Delay / Jitter:*** Atraso / Variação do atraso que os pacotes sofrem;
- ***Errors:*** taxa de erros a que os fluxo são sujeitos.

5.2.4.4 Interface entre BB's (QoSBokerInterface)

Quando um handover precisar de ocorrer, os BB's responsáveis pelo oAR e pelo nAR devem trocar algumas mensagens. O BB responsável pelo oAR deve pedir ao BB responsável pelo nAR a autorização para que o MT possa executar o handover. Essa mensagem é chamada de ***HandoverRequest***. A informação que é enviada nessa mensagem chama-se de ***HandoverResource*** e é constituída pela seguinte informação:

- ***nCoA***: é o novo CoA construído pelo MT através da informação recebida do nAR.
- ***NVUP***: o perfil do utilizador no domínio;
- ***ServiceList***: lista de serviços em uso pelo MT no oAR. O BB vai usar esta informação para configurar o nAR.

Ao contrário do que acontece durante um handover dentro do domínio de um BB, quando o handover acontece entre domínio de BB's distintos é necessário o envio da informação relativa ao NVUP e a descrição dos serviços que o utilizador tem subscritos.

5.2.4.5 Interface do *Radio Gateway*

A interface do Radio Gateway (RG) serve para o BB poder abrir um canal na RG para que um determinado fluxo vindo do core da rede possa prosseguir até um MT. O BB vai ser avisado da existência desse fluxo, uma vez que o AR que liga ao core da rede vai pedir permissão para conduzir o fluxo para um determinado MT. Quando o BB determina que o endereço destino desse

fluxo está para além de uma RG o BB envia uma mensagem à RG para que esta abra o canal rádio e assim o fluxo possa chegar ao destino.

5.2.4.5.1 Mensagens enviadas

OpenChannel – Mensagem enviada pelo BB para a RG para que esta abra um canal até um MT. A RG possui um temporizador interno que de tempos a tempos elimina as aberturas de canais que entretanto não tenham sido renovadas.

5.2.4.5.2 Formato dos dados trocados entre o BB e Radio Gateway

- ***qosb_radio_ bearer_info_t***:
  - ***dscp:*** código dscp do fluxo;
  - ***radio_QdS_class:*** código da classe radio do canal;
  - ***status:*** indica se a mensagem é enviada para abertura de canal por um fluxo existente, ou se se trata de um FHO para a rede da RG.

- ***radio_access_allocation_msg_t:***
  - ***nb_entries:*** define o número de elementos ***qosb_radio_ bearer_info_t*** enviados;
  - ***home_Addr:*** endereço do MT ligado à RG;
  - vários ***qosb_radio_ bearer_info_t:*** lista de elementos que definem os parâmetros do canal a ser aberto.

5.2.4.6 Interface COPS_RSVP

A interface COPS_RSVP foi desenvolvida com o objectivo de poder integrar routers Cisco na rede gerida pelo BB. Após a análise da informação disponibilizada pelo fabricante verificou-se que o modo mais fácil de implementar o controlo do router Cisco era através de COPS-RSVP. A Figura 53 mostra um esquema da configuração da rede para o funcionamento do BB nesta configuração.

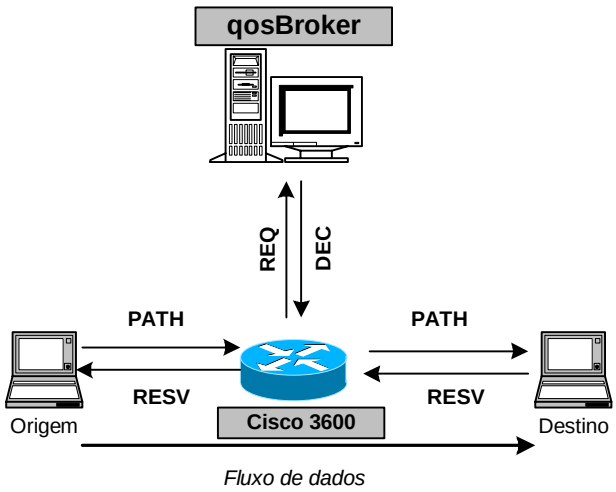

Figura 53 - Funcionamento do BB com router Cisco através de COPS-RSVP

A implementação COPS-RSVP da Cisco foi desenvolvida para IPv4. Este novo interface foi desenvolvido com base no suporte previamente desenvolvido para os routers Moby Dick.

O interface do router contem um módulo chamado de ***CiscoInterface*** que liga o BB aos routers Cisco. O *CiscoInterface* possui dois módulos internos:

- ***CiscoSocket***: consiste numa *thread* que escuta os pedidos COPS em IPv4 por onde é feita a comunicação com os routers Cisco;
- ***CiscoParser***: trata-se de um conjunto de funções que lêem os objectos COPS presentes nos pedidos enviados pelos routers.

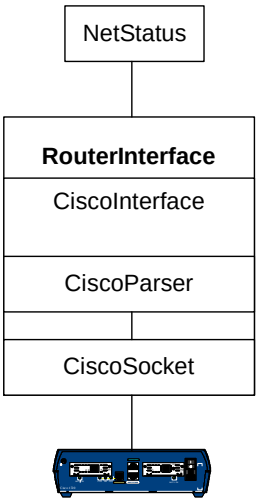

Figura 54 - Suporte COPS-RSVP para routers Cisco

A Figura 54 ilustra a composição do módulo CiscoInterface. Na implementação COPS da Cisco verificou-se que a informação referente aos pedidos RSVP que são recebidos pelos routers é encapsulada por eles dentro de um objecto do tipo *Client Specific Info* sem que seja feito qualquer processamento.

5.2.4.7 Interface com o Administrador de Sistema

A interface com o Administrador de Sistema foi concebida com o objectivo de dar informação ao operador humano que faz a manutenção do BB do estado da rede. Com o objectivo de permitir o acesso a esta informação sem que seja necessário uma deslocação para junto da máquina o interface foi desenvolvido sob a forma de uma página Web permitindo o acesso à informação de qualquer máquina que tenha um browser. Foram usados *scripts php* para aceder à base de dados *MySQL* e assim publicar a informação lá armazenada.

A informação fornecida na interface do administrador divide-se em duas partes: a lista de reservas dos serviços em uso no momento, e a lista dos utilizadores registados na rede, com a respectiva lista de serviços para os quais têm autorização de utilizar.

A Figura 55 mostra uma captura de uma página onde se pode ver a lista das reservas activas no BB num dado instante. Como se pode verificar a lista possui o endereço de origem, de destino, o código DSCP, o período de validade e o nome do router de acesso de cada uma das reservas.

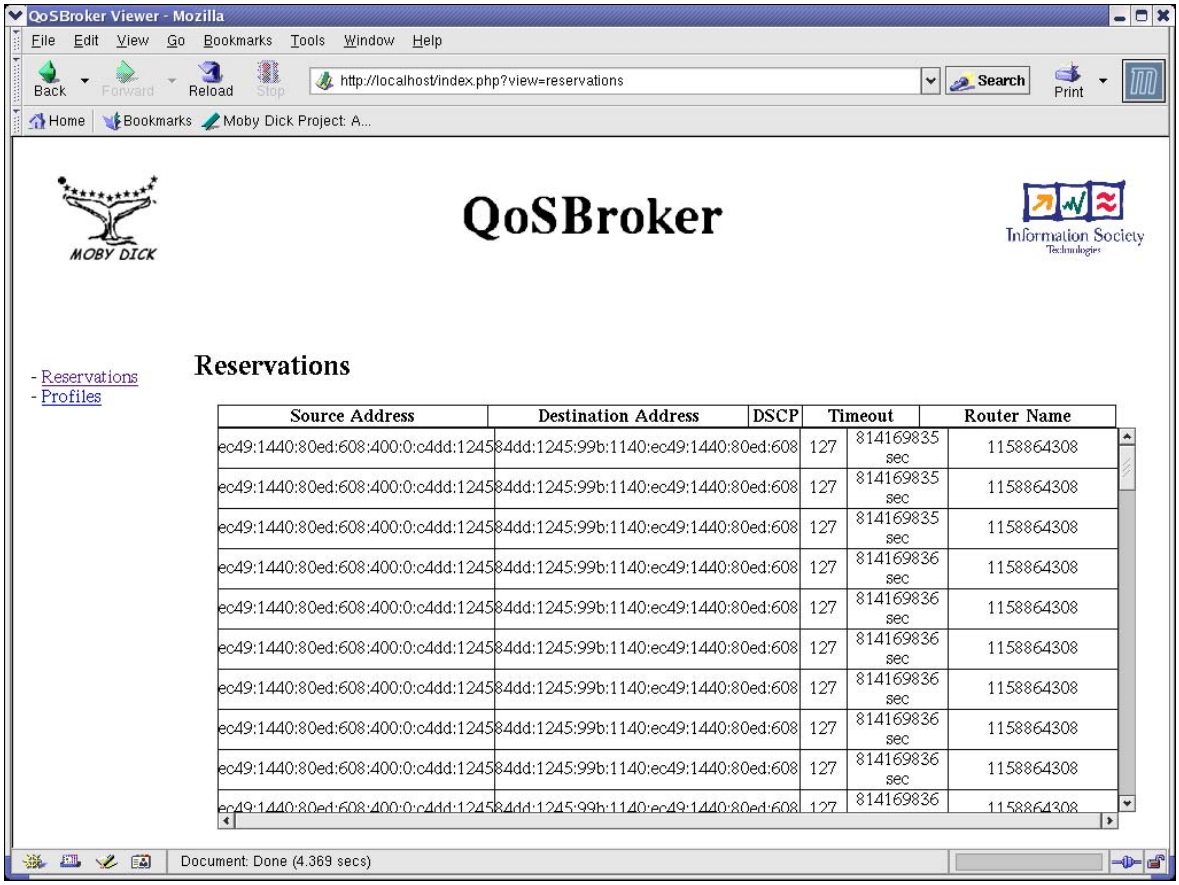

Figura 55 - Imagem da página que mostra as reservas existentes

A Figura 56 mostra uma captura da imagem do interface onde se pode visualizar a lista dos perfis dos utilizadores registados na rede. Como se pode ver da imagem, a lista contém o CoA de cada um dos utilizadores registados na rede bem como a lista dos serviços que cada um dos utilizadores está autorizado a utilizar. A descrição dos serviços contém o endereço de origem, de destino, o código DSCP e o período de validade de cada perfil.

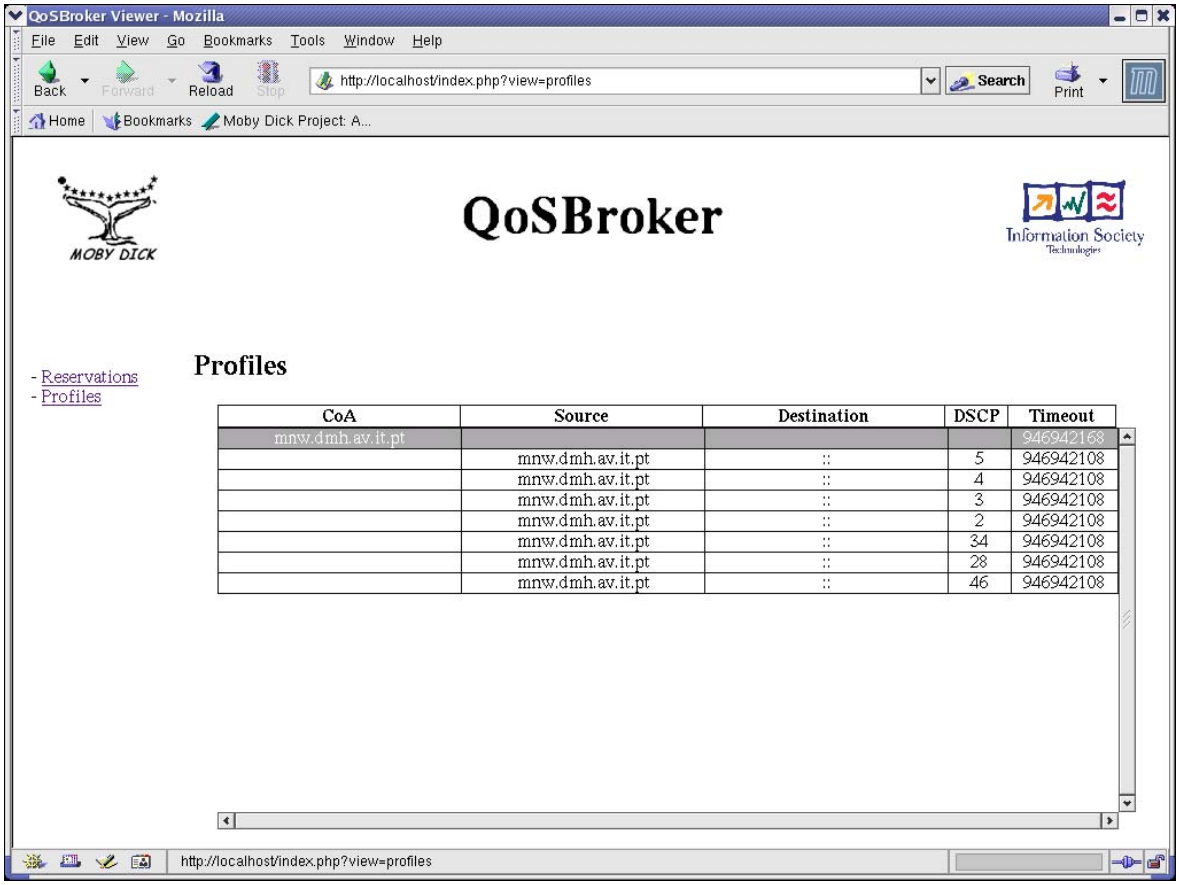

Figura 56 - Imagem da página dos serviços registados

5.2.4.8 Sumário dos interfaces

A Tabela 4 sumaria toda a informação dos interfaces, as mensagens, a altura em que são trocadas e os dados enviados nessas mensagens:

Interface	*Contexto*	*Mensagem*	*Dados enviados*
AAAC	No arranque do BB	SendQoSProfile	ServicesDescription
	No registo	AuthorizeProfile	NVUP
	Anulação do registo	DeAutorizeProfile	
	Em qualquer altura	ValidateNVUP	
		RequestNVUPValidation	
AR	Antes de funcionamento	ConfigRequest	RouterConfig
		ChangeConfig	
	Durante o funcionamento	ResourceRequest	Resource
		ResourceRelease	
	Antes do Handover	HandoverRequest	HandoverResource
NMS	Em qualquer altura	SendQoSConfig	QoSResourceData
BBInterface	Sinalização de processo de handover	RequestHandover	HandoverResource
Radio Gateway	Na chegada de fluxo que tenha como origem ou destino a Radio Gateway	OpenChannel	radio_access_allocation_msg_t

Tabela 4 - Sumário dos interfaces, mensagens e parâmetros

5.2.5 Bases de dados

O BB contem várias bases de dados. Servem para manter armazenados os dados que descrevem o estado e a topologia da rede, bem como dos perfis dos clientes que entretanto foram autorizados pelo servidor de AAAC.

O Sistema de Gestão de Bases de Dados escolhido para o projecto foi o MySQL, que irá ser acedido através de uma API desenvolvida para o efeito.

5.2.5.1 NetworkDB

A base de dados *NetworkDB* vai conter informação que descreve toda a rede que compõem o domínio do BB, bem como a informação necessária à configuração desses elementos de rede.

A Figura 57 ilustra a estrutura da base de dados NetworkDB.

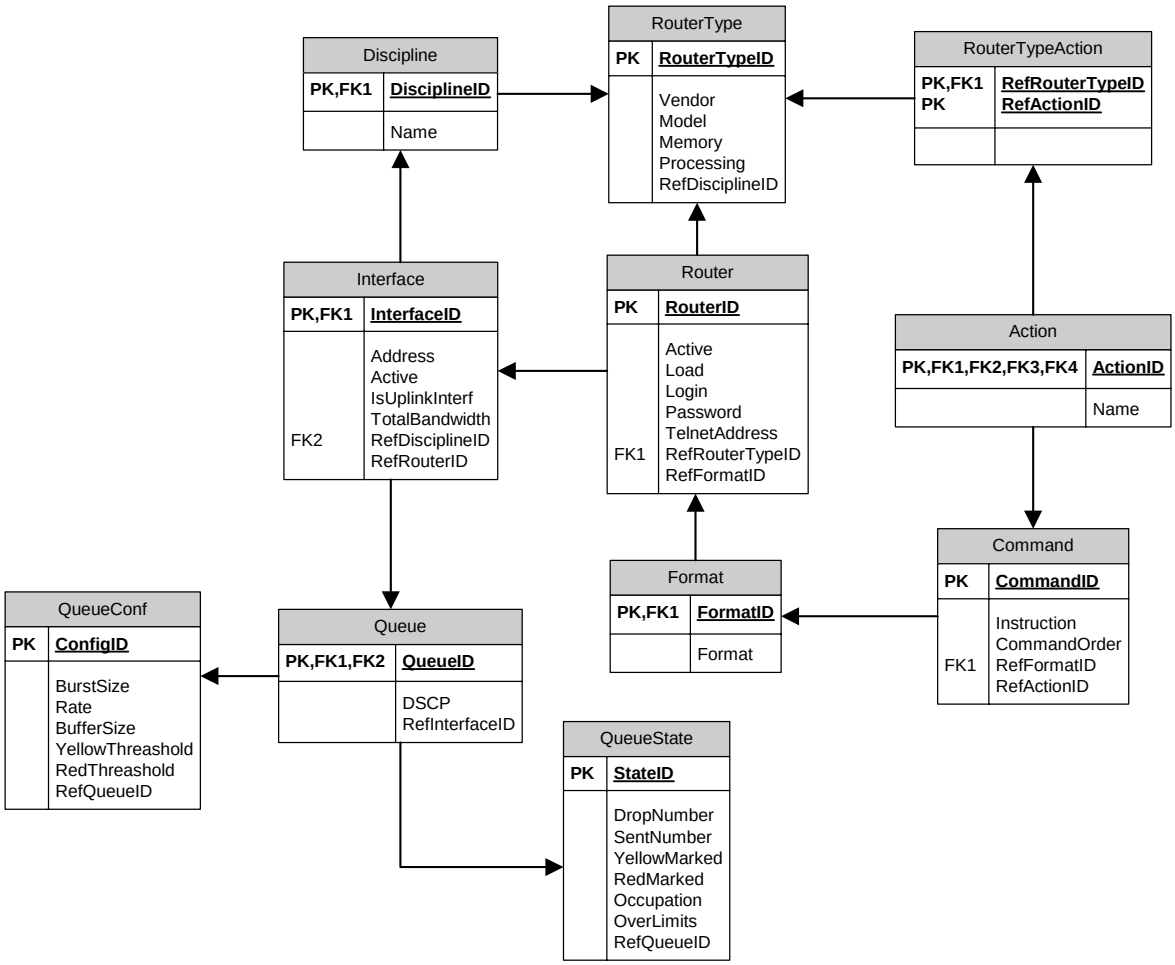

Figura 57 – Diagrama da base de dados NetworkDB

Contem as seguintes tabelas:

- ***Discipline*** - dicionário de disciplinas de escalonamento disponíveis;
- ***RouterType*** - dicionário de tipos de router existentes com as suas características. Contém os seguintes campos:
    - ***Vendor:*** fabricante do equipamento;
    - ***Model:*** modelo do esquipamento;
    - ***Memory:*** memória que o referido equipamento possui;
    - ***Processing:*** velocidade de processamento do equipamento;
    - ***Action:*** dicionário de acções de configuração dos equipamentos;
- ***Command*** - : lista de comandos a dar a um equipamento em ordem para o configurar. Contém os seguintes campos:
    - ***Instruction:*** comando a ser dado ao equipamento;

  - ***CommandOrder:*** número que define a ordem pela qual os comandos devem ser dados de modo a executar uma determinada acção de configuração;
  - ***Format:*** dicionário de formatos de comunicação com os equipamentos;
- ***Router*** - lista de equipamentos que pertencem à rede que constitui o domínio do BB. Possui as seguintes propriedades:
  - ***Active:*** flag indicativa do estado do equipamento;
  - ***Load:*** valor numérico indicativo do índice de sobrecarga do equipamento;
  - ***Login:*** nome do login de gestão do equipamento;
  - ***Password:*** *password* do login de gestão do equipamento;
  - ***TelnetAddress:*** endereço da interface que permite a gestão remota do equipamento;
- ***Interface*** - lista das interfaces de rede disponíveis pelo equipamento. Possui as seguintes propriedades:
  - ***Address:*** endereço da interface;
  - ***Active:*** variável indicativa do estado da interface;
  - ***IsUplinkInterf:*** define se a interface serve de interface para o core da rede ou se não;
  - ***TotalBandwidth:*** define a largura de banda disponível à interface;
- ***Queue*** - lista de filas de espera que cada interface possui. Possui as seguintes propriedades:
  - ***Size:*** tamanho da fila de espera;
  - ***Occupation:*** taxa de ocupação da fila de espera;
  - ***DSCP:*** código indicativo da classe de serviço a que a fila de espera está alocada;
  - ***Bandwidth:*** Largura de banda que está alocada ao serviço desta fila de espera;
  - ***MinBandwidth:*** mínimo de largura de banda que tem que ser disponibilizada à interface após o *service stealing*;
  - ***Priority:*** prioridade que o escalonador atribui ao serviço desta fila;
- ***QueueState –*** mantém os valores que representam o estado das filas:
  - ***StateID:*** chave da tabela de estado das filas;
  - ***DropNumber:*** Número de pacotes descartados***;***
  - ***YellowMarked:*** Número de pacotes marcados de amarelo***;***
  - ***RedMarked:*** Número de pacotes marcados de vermelho***;***
  - ***Occupation:*** Ocupação da fila***;***

- ***QueueConf*** – mantém um conjunto de valores que representam a configuração actual das filas:
    - ***ConfID:*** chave da tabela de configurações das filas***;***
    - ***BurstSize:*** Tamanho de Burst***;***
    - ***Rate:*** Taxa de transferência da queue***;***
    - ***BufferSize:*** Tamanho de Burst***;***
    - ***YellowThreashold:*** Limite de marcação para amarelo***;***
    - ***RedThreadshold:*** Limite de marcação de vermelho***;***

5.2.5.2 UserProfileDB

A base de dados *UserProfileDB* armazena a informação relacionada com a descrição dos serviços no domínio do sistema AAAC, o perfil dos utilizadores autenticados e dos serviços que os mesmos estão autorizados a usar. A Figura 58 mostra um diagrama da base de dados.

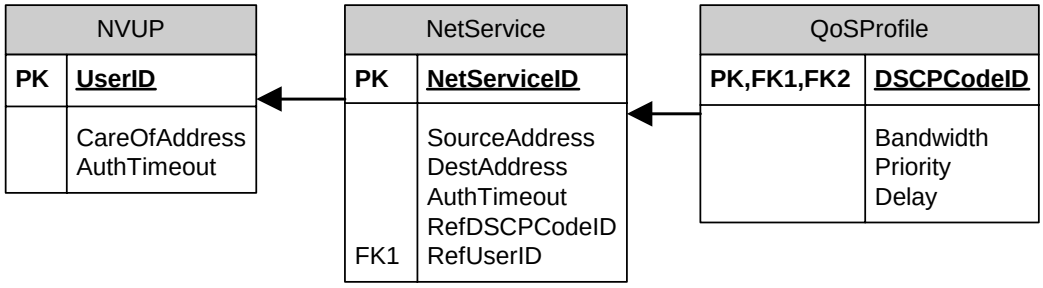

Figura 58 – Diagrama da base de dados UserProfileDB

Quando o BB inicia a sua operação o servidor de AAAC começa por lhe enviar a definição dos parâmetros de QdS para cada um dos serviços de rede disponíveis para essa rede. Essa informação é inserida na tabela *QoSProfile*. A partir daí, o BB consulta esta tabela cada vez que precisa de saber quais as características de um determinado serviço. Quando o AAAC pretende fazer o registo de um utilizador envia uma mensagem de AuthorizeProfile para o BB, com a definição do NVUP e dos serviços que estão subscritos pelo utilizador. Essa informação é colocada respectivamente nas tabelas *NVUP* e *NetService*. A base de dados contém assim as seguintes tabelas:

- ***NVUP*** - contém os perfis dos utilizadores que estão autenticados. É constituída pelos seguintes campos:
    - ***UserID:*** handle que serve para destinguir os perfis dos utilizadores nas mensagens de revalidação e remoção dos perfis;

- ***CareOfAddress:*** endereço origem em uso no momento actual;
- ***AuthTimeout:*** prazo de validade do perfil.

- ***NetService*** - serviço de rede que o utilizador está autorizado pelo AAAC a usar. É constituída pelos seguintes campos:
  - ***DestAddress:*** endereço destino do serviço de rede;
  - ***SourceAddress:*** endereço origem dos pacotes que compõem o fluxo deste serviço;
  - ***AuthTimeout:*** prazo de validade da autorização de utilização do serviço;
  - ***RefDSCPCodeID:*** código de DSCP que indica qual a entrada da tabela ***Services*** que descreve o serviço autorizado.

- ***QoSProfile*** - lista dos serviços disponibilizados no domínio do sistema de AAAC. Contém os seguintes elementos:
  - ***DSCPCodeID:*** código de DSCP que identifica os parâmetros de qualidade de serviço de cada perfil;
  - ***Bandwidth:*** largura de banda disponível no perfil de qualidade de serviço;
  - ***Priority:*** prioridade dada aos pacotes de um fluxo pertencente ao perfil de qualidade de serviço;
  - ***Delay:*** atraso máximo permitido aos pacotes que pertençam a este perfil de qualidade de serviço;
  - ***DestAddress:*** endereço destino dos pacotes que pertençam ao perfil.

Esta base de dados será colocada em memória devido aos motivos que a seguir se referem:

- A Quantidade de informação destas tabelas é relativamente pequena (alguns milhares de bytes), não constituindo grande problema para a máquina mante-la em memória;
- A sua informação será extremamente volátil: a informação que se irá manter valida durante mais tempo será a da tabela QoSProfile que durará enquanto o BB correr. Toda a outra informação tem um prazo de validade de alguns segundos. Apesar de uma das tabelas durar toda a sessão do BB essa tabela ocupa somente umas dezenas de bytes, não constituindo por isso nenhum tipo de problema o espaço de memória por si ocupada;
- A frequência de consulta desta informação é enorme sendo bastante importante a rapidez de consulta desta informação para o desempenho global do BB;

Optou-se assim por criar uma classe que representa esta base de dados e utilizar classes de STL (Standard Template Library) para alocar e gerir o array de estruturas que constituem as tabelas.

5.2.5.3 NetStatusDB

A base de dados NetStatusDB é uma base de dados que serve para manter em memória o estado da rede, e assim facilitar que as decisões de controlo de admissão sejam efectuadas de um modo rápido. Tal como aconteceu com a base de dados *UserProfileDB* decidiu-se criar uma classe que substituísse a base de dados com objectos de classes STL de modo a que se mantivesse e gerissem os dados em memória. Na altura em que o BB arranca, o módulo NetStatus é responsável pela leitura dos dados que existem na base de dados *NetworkDB* e por os replicar nesta base de dados. Ao longo do funcionamento do BB o módulo NetStatus encarrega-se de inserir na *NetStatusDB* os dados referentes às decisões de controlo de admissão que toma, bem como os dados provenientes dos módulos de monitorização do estado da rede como *NetProbe* e *RouterInfo*. Parte desta informação é volátil só sendo válida durante o tempo de operação do BB, de modo que não faz sentido inserir estes dados novamente na base de dados NetworkDB. A estrutura da base de dados NetworkDB encontra-se ilustrada na Figura 59.

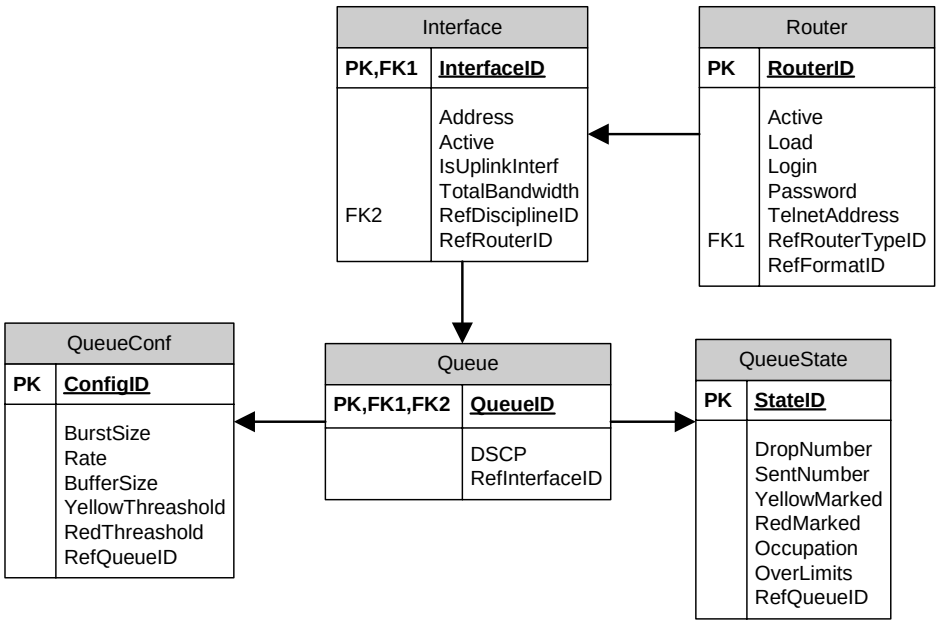

Figura 59 – Diagrama da base de dados NetStatusDB

A base de dados contém as seguintes tabelas:

- ***Interface***: lista de interfaces disponíveis em cada router;

- ***Router***: lista de routers existentes no domínio gerido pelo BB;
- ***Queue***: lista de filas de uma interface;
- ***QueueState***: Estado de configuração da fila;
- ***QueueConf***: Dados de configuração da fila.

5.3 TESTES DO SISTEMA

Os testes executados ao sistema pretenderam analisar dois aspectos principais:

- o desempenho do programa no que se refere a tempos de resposta às solicitações dos outros elementos de rede;
- a requisitos computacionais por ele exigidos.

5.3.1 Rede de Testes

A rede de testes local usada é ilustrada pelo esquema da Figura 60, onde se apresentam as máquinas com os endereços dos vários interfaces, os seus nomes e com a sua função na rede. O Broker foi também estado em *trial sites* do projecto *Moby Dick* em Madrid, Estugarda e Nice.

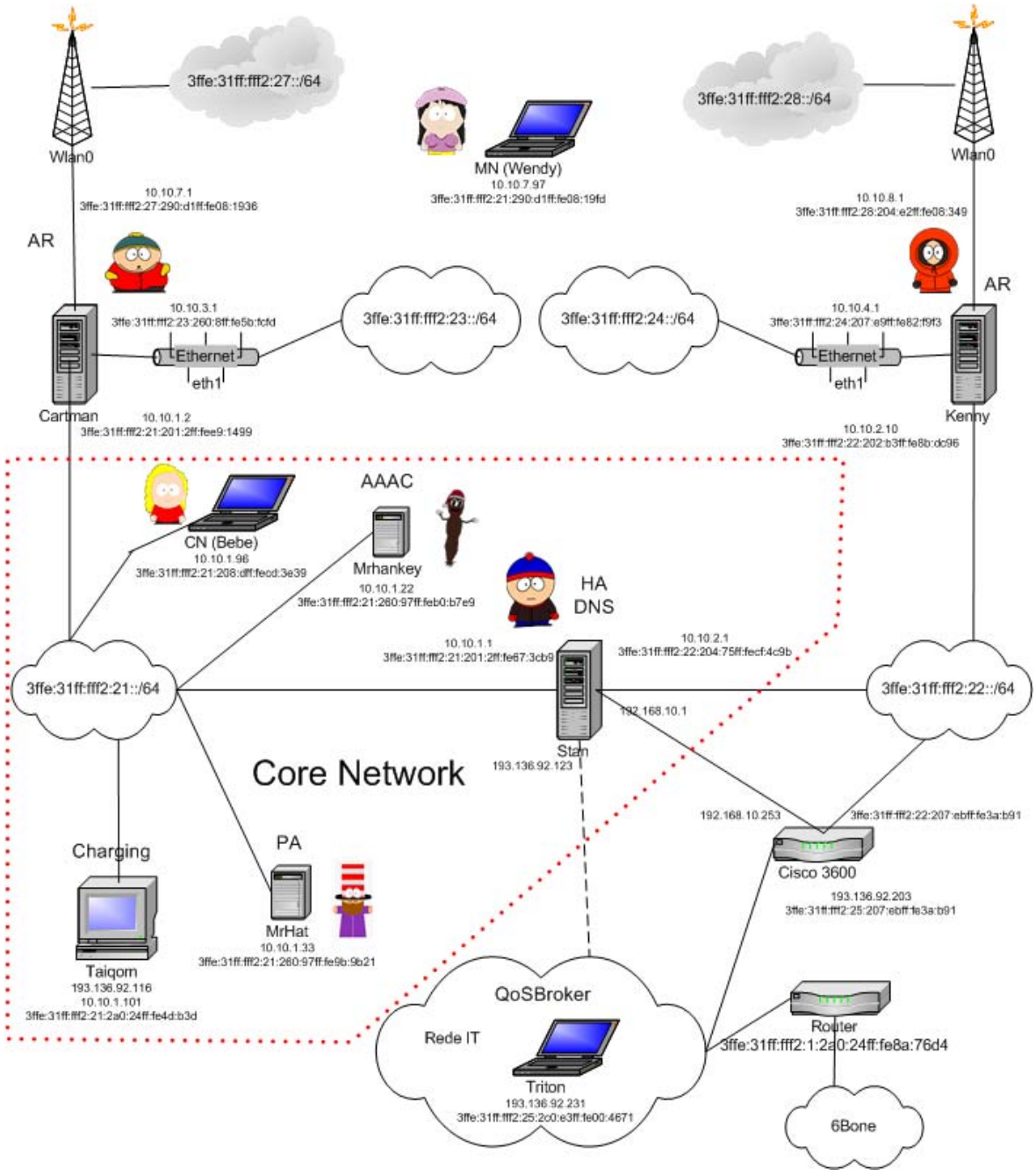

Figura 60 - Rede de testes Moby Dick do Instituto de Telecomunicações de Aveiro

A Tabela 5 mostra a descrição das máquinas que compõe a rede de testes, com o seu nome, a descrição do hardware da máquina, a sua função na rede bem como a distribuição do sistema operativo.

Nome da Máquina	*Descrição da Máquina*	*Entidade Atribuída*	*Distribuição do Sistema Operativo*
Stan	AMD Duron 1.3 GHz, 128 MB Mem, 40 GB HDD	Home Agent/DNS	Suse 8.0
MrHankey	Intel Pentium 200 MHz MMX, 160 MB Mem, 6 GB + 3 GB HDD	AAAC Server	Redhat 7.2
Cartman	Intel Pentium II 266 MHz, 320 MB Mem, 40 GB HDD	AR/QdS Router/AAA Attendant	Suse 8.0
Kenny	Intel Pentium 4 2.4 Ghz, 512 MB Mem, 120 GB HDD	AR/QdS Router/AAA Attendant	Suse 8.0
Kyle	2x Intel Pentium III 1000 Mhz, 256 MB Mem, 40 GB HDD	BB/Charging&Accounting	Redhat 8.0
MrHat	Intel Pentium Pro 200 MHz, 96 MB Mem, 1GB + 2 GB + 3 GB HDD	Paging Agent	Suse 8.0
Wendy	Intel Pentium 4 2.6, 256 MB Mem, 30 GB HDD	Mobile Node	Suse 8.0
Bebe	Intel Pentium 4 2.6, 256 MB Mem, 30 GB HDD	Correspondent Node	Suse 8.0
Taiqom	Intel Pentium 4 2.4 Ghz, 512 MB Mem, 80 GB HDD	BB	Suse 8.2
Triton	Intel Celeron 266 Mhz, 196 MB Mem, 6 Gb HDD	BB	RedHat 7.3
Ike	Intel PII 350, 128 MB Mem, 4Gb HDD	AR/QdS Router/AAA Attendant	RedHat 7.2

Tabela 5 - Descrição das máquinas que compunham a rede de testes

5.3.2 Tempos de resposta

Um dos parâmetros mais importantes do desempenho de um servidor é o tempo de resposta às solicitações que lhe são feitas. Nos testes realizados ao BB foram medidos os tempos de resposta aos vários pedidos. Os resultados são apresentados nesta secção.

5.3.2.1 Mensagens entre AR e BB

Para medir os tempos de resposta às solicitações do AR foram feitas medidas com o BB a correr no *Taiqom* e usados os AR *Kenny* e *Cartman*.. A Tabela 6 mostra o resultado desses testes em média.

Mensagem	*Tempo de resposta em µs*
Client-Open	81
Pedido de Configuração	54320
Recusa de Pedido acesso	733
Aceitação de pedido de core	1537
Aceitação de pedido de acesso	3857
Keep-Alive	140
FHO	17746

Tabela 6 - Tempos médios de resposta às solicitações do AR

As conclusões mais interessantes retiradas da análise destes resultados são as seguintes:

- O tempo de aceitação de um novo cliente é extremamente rápido (81 µs) o que era esperado pelo facto de ser uma acção muito simples, só necessitando o BB de criar uma entrada na tabela dos AR ligados antes de poder responder;
- O tempo de resposta ao pedido de configuração é o mais elevado de todos os tempos que foram medidos. Isso deve-se ao facto de o BB necessitar de procurar qual o tipo de router em questão, procurar na base de dados qual a configuração que lhe deve entregar, compor uma mensagem COPS com esses valores e enviar ao AR. Esse valor não representa qualquer problema, uma vez que a configuração completa do AR só é pedida faz quando este arranca;
- A recusa de um pedido é mais rápida do que a aceitação uma vez que quando o BB recusa um pedido não necessita reservar os recursos porque não há atribuição;
- A resposta aos pedidos do core é mais rápida do que a pedidos do interface de acesso, porque de acordo com a arquitectura da rede Moby Dick o core é uma zona "privilegiada" da rede, de onde se aceitam todos os pedidos e não se verificam se os pedidos estão de acordo com os perfis dos utilizadores. É de notar que o BB implementa outras políticas mais complexas, se desejado;
- O tempo de resposta a um pedido de FHO é alto, podendo comprometer a resposta da rede ao movimento de um MT. Idealmente segundo os requisitos do projecto *Moby Dick* o tempo de resposta de toda a rede deveria ser da ordem do tempo que é actualmente gasto pelo BB. Existem dois motivos para que isto aconteça. Em primeiro lugar a implementação existente usa um vector para armazenar os dados dos routers pertencentes à rede e a lista de *sockets* de comunicação com os routers. Esse tipo de implementação torna lento o processo de procura. No processo de FHO, começa-se por procurar qual o router que tem o endereço indicado no pedido e depois procura-se o socket que existe aberto para se poder enviar a resposta. Isto torna a resposta do BB algo lenta. Em segundo lugar o BB imprime no interface do utilizador dados referentes ao pedido recebido, aos serviços a transferir para o novo AR, bem como a composição do NVUP depois do processo de FHO ter decorrido. A impressão dessa informação é feita de modo a dar informação ao administrador dos serviços que são transferidos e de como está a correr o processo de FHO; por ser muito extensa esta informação adiciona muito tempo à resposta ao pedido de FHO. Em situações em que isto não é feito, o tempo de resposta desce para valores abaixo de 13 ms.

5.3.2.2 Mensagens entre AAAC e BB

Foram também medidos valores dos tempos de resposta das solicitações do servidor de AAAC. A Tabela 7 mostra os valores médios medidos para cada um desses pedidos.

Mensagem	*Tempo de resposta em µs*
Client-Open	84
Definição dos perfis de QdS	384305
Autorização de NVUP	4007

Tabela 7 - Tempos de resposta médios às solicitações do servidor de AAAC

Analisando os valores da tabela podem-se concluir:

- A resposta à mensagem de estabelecimento de ligação tem um valor bastante baixo (84 µs);
- A mensagem de definição dos perfis de QdS contém muita informação que tem que ser lida, interpretada e armazenada pelo BB, o que justifica que tenha uma resposta tão demorada. Não se trata de um problema de implementação, uma vez que esta acção só é executada quando o servidor de AAAC se liga;
- O tempo de resposta ao envio de um NVUP é de 4007 µs, o que parece perfeitamente aceitável.

5.3.2.3 Mensagens entre RG e BB

Entre a RG e o BB existe sempre um AR que intervém no processo de pedido de recursos, e que despoleta o pedido ao BB. Esse AR necessita ainda de uma resposta do BB de modo a que possa conduzir o fluxo em causa. A Tabela 8 mostra as medições dos tempos médios dessas respostas.

Mensagem	*Tempo de resposta em µs*
Contacto da RG	1079
Resposta ao AR	5388

Tabela 8 - Tempos de resposta médios a um pedido de um MT pertencente a uma rede de uma RG

Verifica-se que o tempo de resposta à RG é mais baixo do que ao AR, o que se deve à forma como o BB está implementado. Assim que o BB executa a rotina que faz o controlo de admissão e que decide aceitar um fluxo o BB executa a rotina que determina se a origem ou destino do fluxo a admitido está numa rede de uma RG. Caso isso aconteça o BB envia a mensagem à RG e só depois responde ao AR.

5.3.3 Recursos utilizados

De modo a analisar os recursos consumidos pelo BB foram feitos testes em várias máquinas da rede. Os recursos consumidos foram medidos em duas situações: durante a resposta a um pedido, e durante um pedido de FHO. Devido à baixa utilização do processador mesmo na máquina mais lenta e ao facto de as variações de utilização serem muito rápidas era impossível medir directamente qualquer valor. Foi no entanto possível medir a memória ocupada pelo programa, tanto no momento a seguir ao arranque como depois de estar em funcionamento ligado a todos os elementos do demonstrador. A Tabela 9 mostra o resultado dessa medição.

	Arranque	***Funcionamento***
Código	184	1088
Data + Stack	196	1200

Tabela 9 - Valores de ocupação de memória pelo BB

Com o objectivo de poder fazer uma estimativa da ocupação do processador optou-se por utilizar uma ferramenta que fazia a medição dos valores de utilização e o seu registo em ficheiro. Foi colocada essa ferramenta a correr com o BB na máquina chamada de *Triton* e registados esses valores. Durante o tempo de medição foram executadas as seguintes solicitações ao BB:

- foi iniciado por volta dos 7 segundos;
- foi ligado o servidor de AAAC por volta do segundo 65;
- foi ligado o router *Cartman* por volta do segundo 90;
- o router *Kenny* foi ligado por volta do segundo 100;
- o registo do MT começou por volta do segundo 125 e acabou por volta do segundo 180;
- foi criado tráfego de rede após o segundo 200;
- aconteceu o 1ª FHO depois do segundo 230;
- houve uma actualização do perfil de utilizador depois do segundo 250;
- foi feito um segundo FHO depois do segundo 280;
- houve uma segunda actualização do perfil de utilizador depois do segundo 305;
- foi feito um pedido do AR por volta do segundo 330;
- foi feito o terceiro FHO depois de 360;
- iniciou-se o processo de finalização do BB depois do segundo 375.

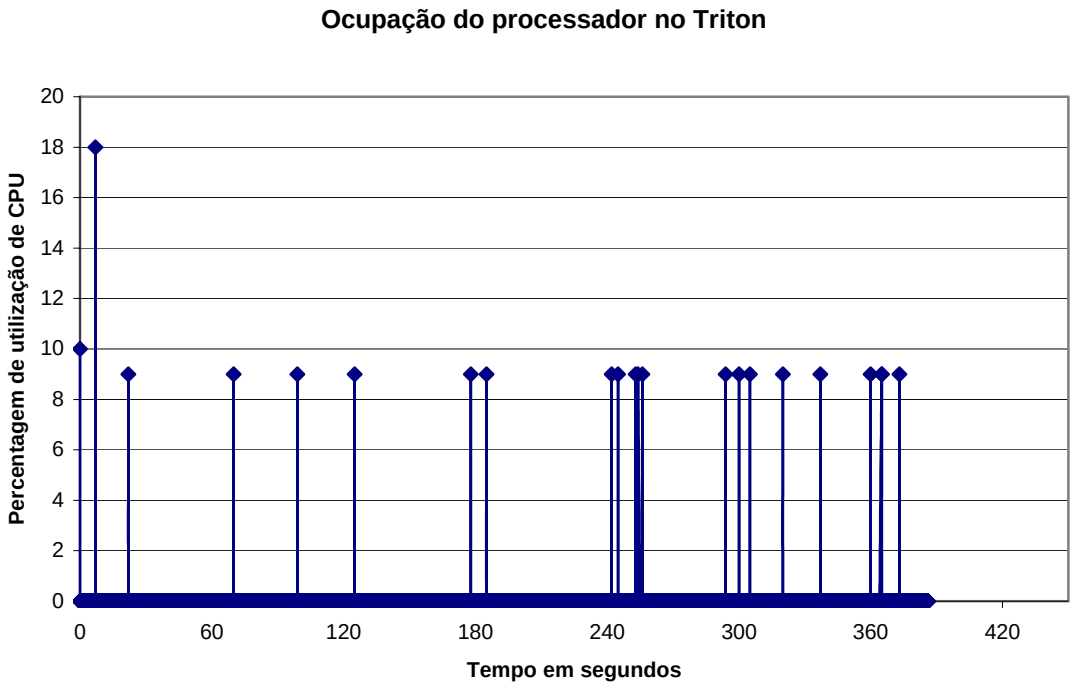

Figura 61 - Gráfico da utilização do processador no Triton

Como se pode verificar pela observação da Figura 61 o valor máximo de utilização do processador verifica-se aos 7 segundos, altura em que este arranca. Pode-se verificar também que os valores de utilização medida são da ordem dos 9% e acontecem sempre que um pedido é feito ao BB por um dos elementos da rede. Os períodos durante os quais a utilização se mantém nestes valores são muito pequenos, notando-se uma maior duração sempre que vários pedidos coincidem.

Atendendo aos baixos valores de utilização do CPU na experiência anterior o mesmo teste foi replicado somente no *Mrhankey* pelo facto de ser a máquina mais lenta existente na rede de testes. Foi traçado um gráfico que pode ser observado na Figura 62.

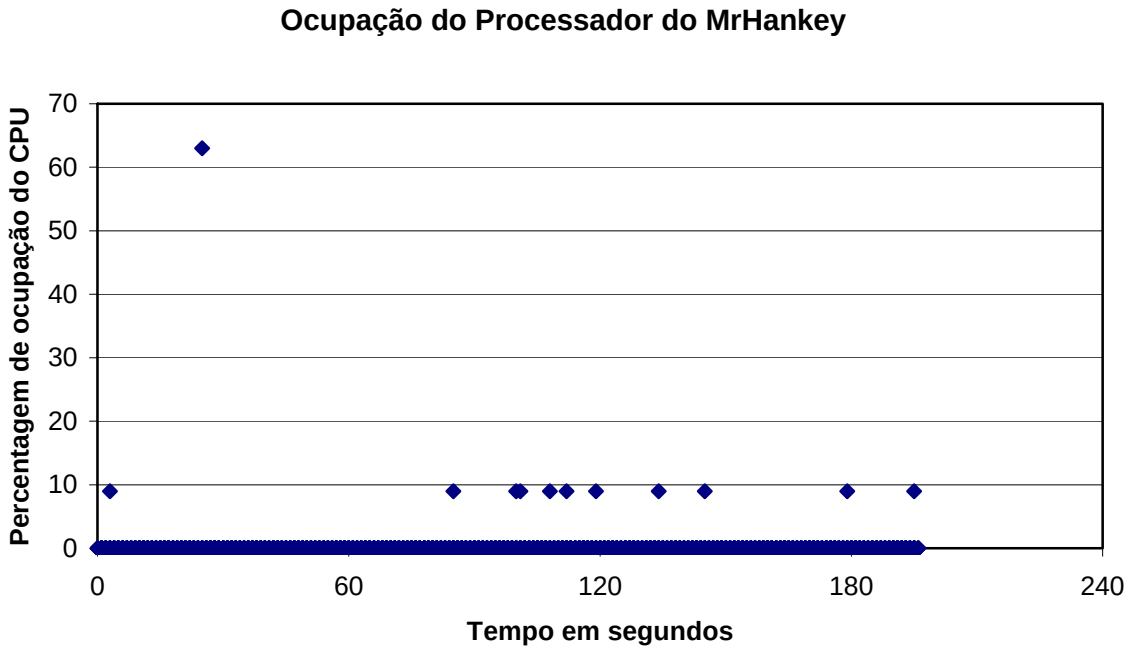

Figura 62 – Gráfico da utilização no MrHankey

Como se pode ver existe uma variação enorme das taxas de utilização do CPU quando o BB arranca, o que se explica com a diferença de capacidade de processamento de ambas as máquinas. Neste teste o pico maior de utilização do CPU correspondem à altura em que a maquina arrancou e é de 63%. As restantes solicitações feitas ao BB traduzem-se em taxas de utilização de 9% o que reflecte a maneira como o sistema operativo calendariza as tarefas do processador.

5.3.4 Base de dados

Por forma a perceber se o tamanho da base de dados se tratava de uma limitação à utilização do BB na missão para que foi desenhado, ou se poderia trazer problemas de escalabilidade, analisou-se o seu tamanho. O tamanho medido foi de 144Kb, o que se explica pela pequena quantidade de máquinas que compõem a rede de testes. Prevê-se no entanto que para uma rede com 100 AR o tamanho da informação presente na base de dados seja de 7,2 Mb, o que não se torna de modo nenhum problemático.

5.3.5 Tempo de arranque

Foi feita uma análise ao tempo necessário ao BB para que esteja disponível para responder a um pedido desde que arranca. Assim que arranca o BB cria as *threads* que escutam as comunicações com os AR e servidor de AAAC, lê a informação que descreve a topologia da rede que está armazenada na base de dados, duplica-a em memória e cria ligações com os outros Brokers da rede. A Tabela 10 mostra os valores que foram medidos para executar essas acções nas diversas máquinas.

Máquina	***Tempo de arranque em us***
Taiqom	15267
Triton	23288
Kyle	13311
Mrhankey	65617

Tabela 10 - Tempos de arranque do BB

Como se pode ver da Tabela 10 o tempo de arranque mais elevado foi medido no MrHankey, cerca de 66 ms, que é uma máquina ultrapassada.

5.3.6 Análise de Profiling

Com o objectivo de estudar as optimizações de que o software desenvolvido podia sofrer utilizou-se uma ferramenta de Profiling. As ferramentas de Profiling correm o programa a testar de modo a poderem monitorizar e registar o seu comportamento.

A ferramenta utilizada foi o *KCachegrind* [51]. Trata-se de uma ferramenta de visualização para os dados de Profiling gerados pelas ferramentas de *Profiling Cachegrind* e *Calltree*. Basicamente estas ferramentas correm o software a testar dentro de um emulador de processador e capturam os dados relevantes acerca do funcionamento do software em teste que depois registam em disco. Os dados registados incluem o número de instruções executadas, acessos à memória de dados, *memory leaks*, utilização de variáveis não inicializadas, entre muitas outras coisas. Tanto quanto é anunciado em [51], a ferramenta de monitorização não influencia o comportamento do software em teste, limitando-se a tornar mais lenta a sua execução cerca de 50 vezes. Trata-se de uma ferramenta poderosíssima uma vez que permite analisar anomalias no comportamento de um programa que por vezes não são fáceis de detectar.

Feita a monitorização do comportamento do programa a testar, o *KCacheGrind* permite a análise da distribuição do tempo gasto pelas diversas funções, criação de gráficos com a distribuição do tempo de processamento e até a visualização do código em questão. Não requer

sequer qualquer tipo de recompilação do código e permite a utilização de librarias e *plugins*. A Figura 63 mostra um detalhe da ferramenta utilizada.

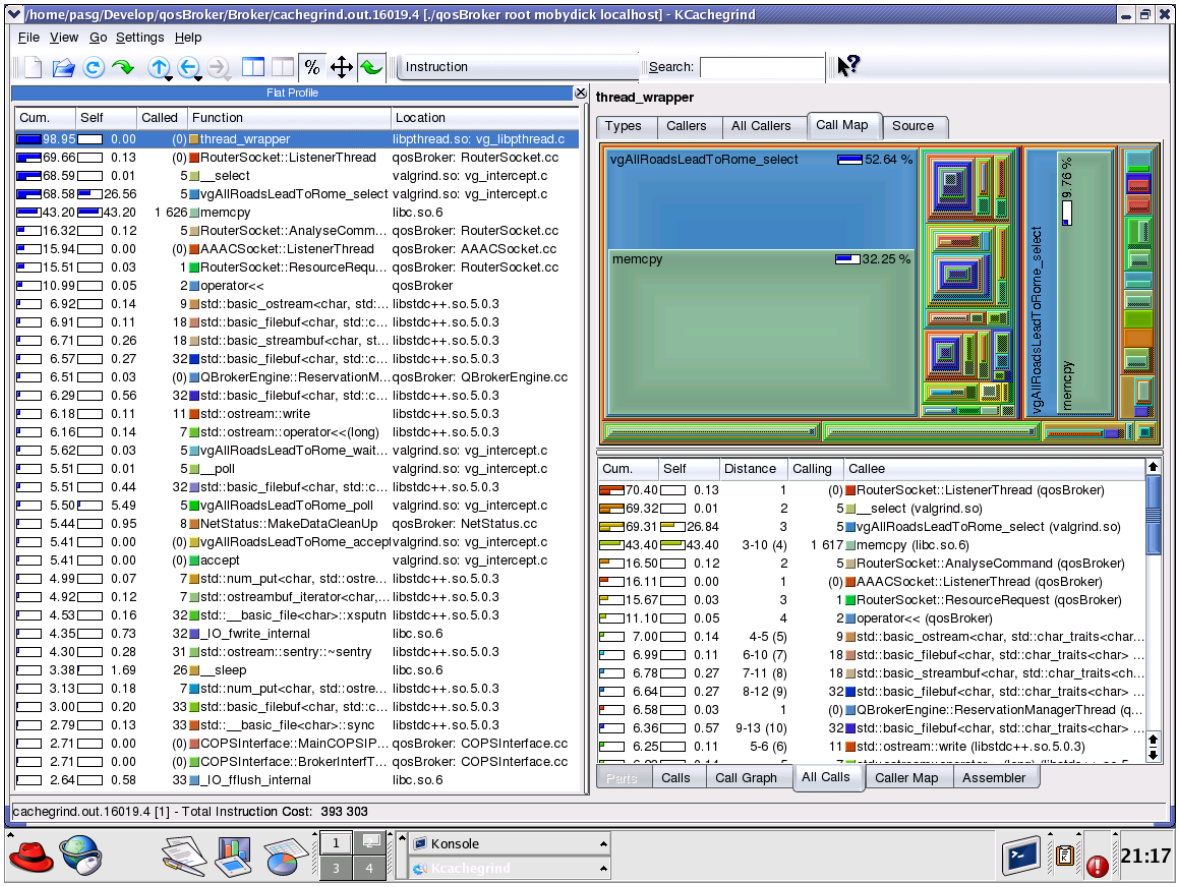

Figura 63 - Captura de uma janela do KCachegrind

5.3.6.1 Análise dos resultados

Feita a monitorização do comportamento do programa, foram analisados os resultados. A Tabela 11 mostra a distribuição do processamento feito pelas diferentes *threads* do BB.

Custo (%)	*Chamadas*	*Função*
55,77	4	RouterSocket::ListenerThread
18,77	2	AAACSocket::ListenerThread
3,70	1	COPSInterface::MainCOPSIPV6InterfThread
3,04	1	COPSInterface::BrokerInterfThread
2,73	1	QBrokerEngine::ReservationThreadManager
1,33	1	QBrokerEngine::AuthorizationThreadManager
0,34	2	MDCLient::ConnectNeighbElement

Tabela 11 - Análise da distribuição do processamento pelas chamadas feitas pelo gestor de threads

Como se pode verificar pela Tabela 11, a *thread* que comunica com os routers de acesso é a *thread* que mais recursos consome, resultado do número de pedidos feitos pelos dois routers de acesso a que estava ligado o BB quando foi executado o teste. Seguem-se a *thread* que comunica com o sistema de AAAC, a *thread* que implementa o servidor COPS, a *thread* do interface com outros Brokers, as *threads* que mantêm as listas de perfis autorizados e lista de reservas. Finalmente aparecem na lista as threads clientes da RG e de outros Brokers.

É de notar que esta distribuição do processamento é consequência da topologia da rede de testes. Numa rede de um operador existem muitos AR, e só um servidor de AAAC, o que irá aumentar ainda mais a percentagem de processamento da *thread* que comunica com os AR.

A Tabela 12 mostra a distribuição do processamento pelas funções chamadas pelas *thread* que implementa a comunicação com os AR.

Custo	*Chamadas*	*Função*
41,94	26	RouterSocket::AnalyseCommand
0,22	26	read_message

Tabela 12 - Distribuição do processamento pelas funções chamadas pelas *thread* que implementa a comunicação com os AR

A grande maioria do recursos são consumidos processando os pedidos dos AR, sendo uma pequena minoria consumida lendo os pedidos.

No que respeita à distribuição do processamento feito pelas funções chamadas por *RouterSocket::AnalyseCommand* de acordo com a Tabela 13 verifica-se que a leitura dos pedidos representa 0,18% e que a maior parte do processamento é feito pela função que analisa os pedidos dos routers.

Custo	*Chamadas*	*Função*
41,69	9	RouterSocket::ResourceRequest
0,18	15	read_report_BB
0,01	1	send_keepalive
0,00	1	ServerSocket:SendDebugUDP

Tabela 13 - Distribuição do processamento das chamadas de RouterSocket::AnalyseCommand

Analisando o a distribuição do processamento feito pelas funções chamadas por *RouterSocket::ResourceRequest* (Tabela 14) verifica-se que a função que consome mais recursos é a função que faz controlo de admissão, seguindo-se as funções que imprimem a informação na consola do programa. De seguida aparecem as funções que interpretam os pedidos do AR, a função de sistema *memcopy* e aparecem novamente na lista funções de impressão de mensagens na consola.

Custo	Chamadas	Função
31,33	5	COPSInterface::ResourceRequest
7,35	18	operator<<(std::ostream&, print_color)
1,66	9	std::basic_ostream<char, std::char_trails<char>>
0,53	9	std::ostream::operator<<operator
0,51	9	std::ostream::operator<<(int)
0,18	9	send_RA_decision
0,06	9	read_request
0,02	9	read_req_AR
0.01	18	Memcpy
0,00	18	print_color::print_color
0,00	9	ServerSocket::SendDebugUDP

Tabela 14 - Distribuição RouterSocket::ResourceRequest

Analisando ainda a distribuição da capacidade de processamento gasto dentro da função que implementa o algoritmo de controlo de admissão (Tabela 15) verificamos que a maior parcela continua a ser gasta pelas funções de *debug* do BB, seguido pela função de verificação dos pedidos e pela função que classifica o pedido em termos do tipo de interface de onde partiu.

Custo	Chamadas	Função
15,76	5	operator<<(std::ostream&, NET_SERVICE)
8,17	20	operator<<(std::ostream&, print_color)
3,68	10	std::basic_ostream<char, std::char_trails<char>>
2,97	5	UserProfile::IsAccessInProfile
0,59	10	std::ostream::operator<<
0,14	5	NetStatus::IsCoreInterface
0,00	20	print_color::print_color

Tabela 15 - Distribuição de COPSInterface::ResourceRequest

Analisando a totalidade dos recursos gastos pela *thread* que comunica com AR verificamos que a maior parcela é gasta por funções de impressão de mensagens na consola o que representa mais de 38% do processamento total do BB. A Figura 64 mostra um gráfico com essa distribuição e permite-nos concluir que a esmagadora maioria do processamento gasto é feito com a impressão de mensagens no Interface gráfico. Como essas mensagens só são impressas quando se utiliza uma compilação do programa em modo de *debug*, esse modo deve ser evitado pois tem implicações sérias na performance do software.

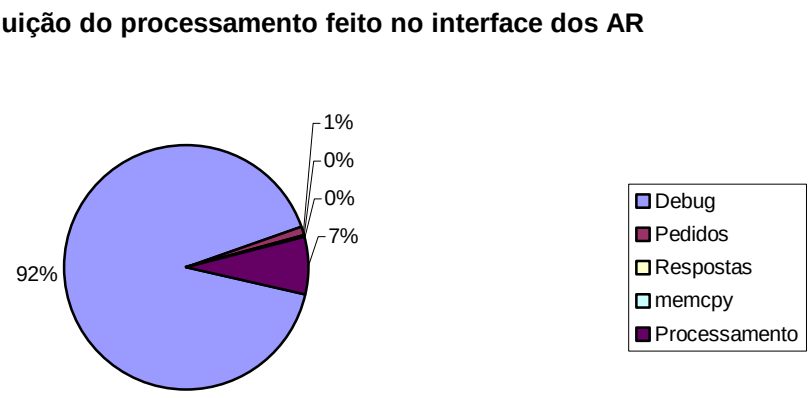

Figura 64 - Gráfico de distribuição do processamento feito no interface do AR

Pode-se ainda verificar que o processamento de pedidos é a segunda maior fatia o que se explica com o mecanismo de controlo de admissão que o BB implementa. O gráfico da Figura 65 mostra a distribuição do tempo de processamento.

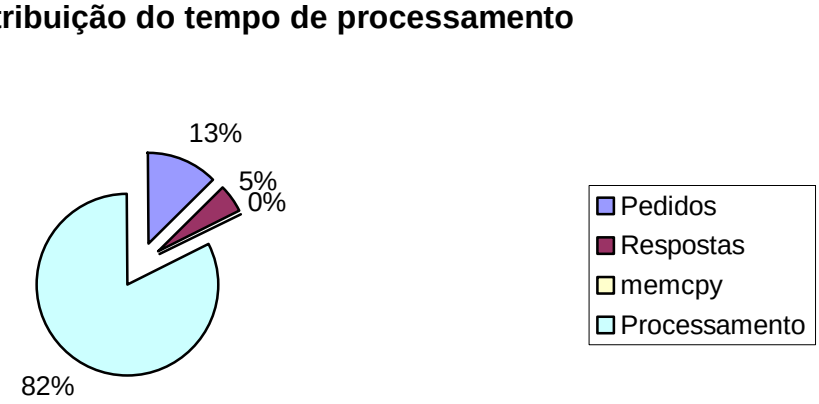

Figura 65 - Distribuição do tempo de processamento sem considerar a componente de debug

Fazendo a mesma análise em relação à *thread* do servidor de AAAC verifica-se a maior parte do processamento é feita a processar os pedidos do servidor de AAAC.

Custo	*Chamadas*	*Função*
26,92	1	AAACSocket::AnalyseCommand
0,01	1	read_message

Tabela 16 - Distribuição do processamento das chamadas feitas por AAACSocket::ListenerThread

Ao analisar a distribuição do processamento feito pela rotina que processa os pedidos de autorização (Tabela 18) verifica-se que a maior fatia é gasta pela função *AAACSocket::AuthorizeProfile* e que a rotina que lê a informação de definição dos perfis tem uma percentagem desprezável.

Custo	*Chamadas*	*Função*
26,92	1	AAACSocket::AuthorizeProfile
0,00	1	MiscUtils::ReadReport
0,00	1	ServerSocket::SendDebugUDP

Tabela 17 - Distribuição do processamento das chamadas de AAACSocket::AnalyseCommand

Verificando também a distribuição do processamento gasto pelas funções chamadas por *AAAC::AuthorizeProfile*, como se pode verificar pela Tabela 18, verifica-se que a maior fatia de processamento é gasto pela função que imprime o conteúdo dos NVUP registados na consola do programa, seguido pela função que faz o registo das autorizações recebidas.

Custo	*Chamadas*	*Função*
22,55	1	operator<<(std::ostream&, ST_NVUP)
3,33	1	COPSInterface::AuthorizeProfile
0,82	2	operator<<(std::ostream&, print_color)
0,16	1	MiscUtils::ReadReqAAAC
0,05	1	COPSInterface::AddConnection

Tabela 18 - Distribuição do processamento das chamadas de AAAC::AuthorizeProfile

No caso das funções chamadas pela função *COPSInterface::AuthorizeProfile* verifica-se que a função que insere na base de dados *MySQL* a informação dos perfis autorizados é a função que mais processamento consome. Verifica-se ainda que a segunda parcela é consumida pela função que faz a comparação de endereços.

Custo	*Chamadas*	*Função*
2,94	1	UserProfile::LogAuthorization
0,40	1	MiscUtils::AreAddressesEqual
0,09	1	UserProfile::AuthorizeServices
0,01	1	Memcpy

Tabela 19 - Distribuição de processamento das chamadas de QBrokerEngine

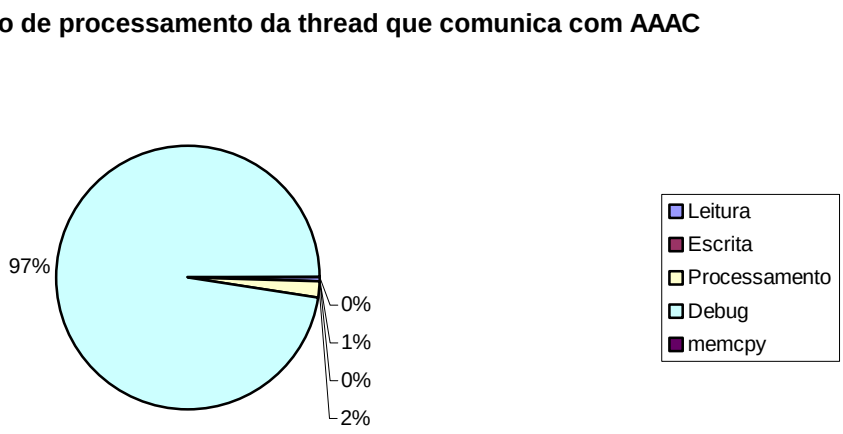

Figura 66 - Gráfico de distribuição do processamento da thread do AAAC

Como se pode constatar do gráfico da Figura 66 a maior parte do processamento feito está relacionado com operações de impressão de informação de funcionamento do programa na consola. O processamento dos pedidos representa 2% e a leitura dos pedidos 1%. É de salientar que a parcela relacionada com escrita de mensagens deste interface é nula, uma vez que a comunicação com o servidor de AAAC é unidireccional, por restrições na rede Moby Dick. O gráfico da Figura 67 mostra a distribuição do tempo de processamaneto.

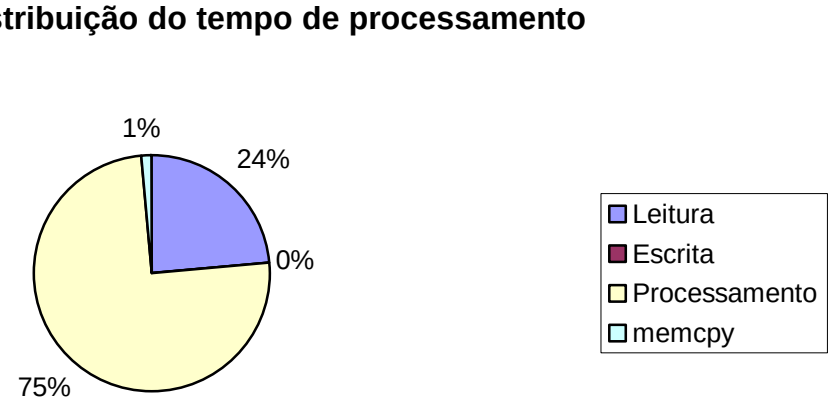

Figura 67 - Distribuição do tempo de processamento sem a componente de debug

Classificando cada uma das funções do BB de acordo com a tarefa que cada uma executa, retirando as parcelas relacionadas com o Interface gráfico podemos analisar como é dividido o processamento do BB. A Figura 68 mostra um gráfico dessa distribuição.

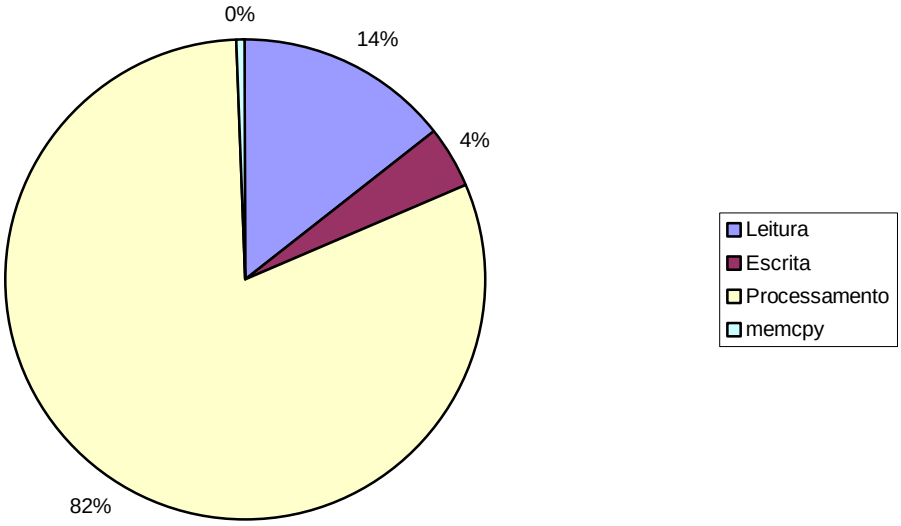

Figura 68 - Distribuição global do processamento em função do tipo de função

Analisando os dados verifica-se que a maior parte do processamento é gasto com acções de processamento, acções como o mecanismo de controlo de admissão, comparação dos serviços pedidos com os perfis autorizados. A leitura de pedidos é a segunda parcela seguida da resposta aos pedidos, o que não constitui também qualquer surpresa uma vez que normalmente as mensagens dos pedidos transportam sempre mais informação do que as respostas e que existe um interface com o BB que é unidireccional.

5.3.7 Análise de escalabilidade

O número de máquinas existentes no demonstrador não permite que sejam feitos testes de escalabilidade. Pode-se no entanto fazer uma previsão do comportamento do BB numa rede de um operador de telecomunicações. Seguem-se cálculos da memória necessária para armazenar os dados necessários ao funcionamento do BB num cenário destes e ainda o calculo do tempo de resposta aos pedidos que a rede lhe faria.

5.3.7.1 Descrição de um cenário

Um cenário possível para um operador de 4G pode ser descrito pelos seguintes pontos:

- 4500 routers de acesso;
- 10000 interfaces de rede;
- 6 milhões de utilizadores;
- 4 serviços por utilizador;
- 10% de utilização simultânea.

Numa rede desta dimensão poderíamos ter 600000 utilizadores registados e 2,4 milhões de reservas em simultâneo, em média.

5.3.7.2 Impacto na memória

A Tabela 20 mostra a memória ocupada por cada um dos itens do cenário em estudo. Como se pode verificar a quantidade de memória necessária para o funcionamento de um BB nestas condições está ao alcance de qualquer PC recente.

Numero de elementos	*Tamanho da estrutura (bytes)*	*Tamanho em memória (Mb)*
4500 routers de acesso	220	0,990
10K interfaces	36	0,360
600K de utilizadores	80	48
2,4M de serviços	52	124,8
2,4M de reservas	60	144
Total		318,15

Tabela 20 - Memória ocupada por cada um dos itens do cenário em estudo

5.3.7.3 Tempo de processamento

Existem duas tarefas que ocupam quase a totalidade do processamento do BB. Tratam-se da resposta a um pedido de reserva e de autorização de um perfil de um utilizador. Pode-se calcular o tempo que uma maquina semelhante à que foi usada nos testes necessitaria para executar estas tarefas no cenário proposto.

Assumindo que as autorizações dos clientes seriam renovadas de 60 em 60 segundos teríamos 10000 renovações por cada segundo. Usando o valor do tempo de resposta a um pedido de autorização de 120 µs o BB levaria 1,2 segundos a processar todos os pedidos de renovação. No caso da renovação dos 2,4 milhões de reservas, com um período de renovação de 60 segundos, teríamos 40000 pedidos de renovação por segundo. Utilizando nos cálculos um tempo de resposta a cada pedido de reserva de 309 µs teríamos que o BB levaria 12,34 segundos a processar todos os pedidos de renovação de reservas (por cada segundo).

5.3.7.4 Análise dos resultados

Analisando os resultados dos cálculos anteriores verificamos que não existe qualquer problema em alojar em memória os dados referentes aos pedidos e topologia da rede descrita no cenário de utilização. Já no que respeita ao tempo necessário para responder aos 4,8 milhões de pedidos por segundo que esta rede faria existiriam grandes problemas, a máquina usada no teste não teria rapidez necessária para o fazer. Pode-se então concluir que uma máquina não é suficiente para processar todos os pedidos do cenário proposto.

No entanto, além de se esperar que numa rede com esta dimensão existissem vários BB's, é de notar que o código encontra-se optimizado para demonstração, não para velocidade, e que máquinas mais recentes são mais rápidas do que as que foram usadas. Existem por isso dois tipos de soluções que podem ser utilizadas para resolver este problema.

Em primeiro lugar pode-se optimizar o BB de modo a que seja mais rápido a processar os pedidos. Essa optimização passava por passar a utilizar *hash-tables* para armazenar a informação em memória. Nesse caso a pesquisa dos objectos em memória deixava de ser linear e a comparação dos elementos necessária à pesquisa deixava de ser campo a campo para ser uma comparação binária de cada elemento.

Seria no entanto insuficiente essa ganho de tempo de resposta, pelo que teria que se recorrer à segunda solução e subdividir a rede em várias subredes cada uma delas gerida por uma instancia de BB. Neste caso seriam necessários mais que uma dezena de BB para gerir por completo a rede.

É importante sublinhar a grandeza destes valores. Tendo 4,8 milhões de pedidos por segundo para uma mesma máquina, considerando o tamanho médio de 250 bytes por pacote, teremos valores de tráfego destinado esta máquina na ordem de 1 Gb. Uma largura de banda deste valor constitui, quer em termos do interface de rede, quer em termos do bus PCI, um sério problema.

A análise feita tem apesar de tudo um valor essencialmente ilustrativo. É de facto importante para que se tenha uma noção da escalabilidade do software desenvolvido e sob esse ponto de vista o detalhe da análise é satisfatório. Não pode no entanto ser usada para fazer uma previsão da real escalabilidade do sistema por falta de rigor da análise. A principal causa da falta de rigor prende-se com o facto de não ser possível determinar como vão crescer os valores de tempo de resposta com o crescimento das reservas, crescimento do número de utilizadores e com o crescimento do número de routers de acesso. Nos cálculos feitos presumiu-se que este crescimento seria linear, o que não será necessariamente verdade. Uma implementação baseada numa lista tem um crescimento exponencial nas procuras, e isso pode ter um efeito tremendo no tempo de resposta, tendo em atenção que se tratam de milhões de elementos presentes na lista. Qualquer estudo de escalabilidade que pretenda apurar valores minimamente próximos da realidade terá que incluir a

medição de tempos de resposta com vários valores de números de utilizadores, de pedidos e de routers. Só assim será possível perceber como poderá crescer o tempo das respostas e extrapolar para valores reais.

6. CONCLUSÕES

Com o crescimento da utilização e o enriquecimento dos serviços que actualmente a Internet tem conhecido torna-se imperiosa a utilização de mecanismos que garantam QdS. Muita investigação tem sido feita acerca deste tema, e encontram-se já várias soluções comerciais para os problemas criados pela necessidade de fornecer garantias ao encaminhamento do tráfego.

Neste trabalho foi investigada a possibilidade de desenvolver um gestor de Qualidade de Serviço para redes heterogéneas congregando tráfego provindo de redes WI-FI, Ethernet e TD-CDMA. A solução encontrada permitiu integrar com sucesso o tráfego das redes entre diferentes tecnologias, sendo a gestão dos recursos da rede efectuada pelo BB, aplicando critérios de controlo de admissão de acordo com o perfil do cliente e assim garantindo os requisitos de QdS definidos no SLA.

Foi ainda desenvolvido suporte de mobilidade entre redes e entre tecnologias. O suporte de mobilidade entre redes desenvolvido usou a aproximação de estabelecer um novo fluxo antes de que a mudança seja feita designado de *FHO*. O processo de FHO é controlado pelo BB, que verifica a disponibilidade de recursos na nova rede de acesso antes de poder aceitar mudança de redes de acesso. O processo de mobilidade entre tecnologias, funciona de um modo muito semelhante, em que o BB verifica a disponibilidade dos recursos na nova rede e envia a decisão de aceitação da mudança de rede de acesso. O suporte de mobilidade foi testado com sucesso, quer entre redes de tecnologias semelhantes, quer entre redes de diferentes tecnologias. Foi verificado que o tempo de resposta do BB é crítico para o sucesso da mudança de rede, no caso de mudança entre redes de arquitecturas semelhantes. Esta limitação deve-se ao facto de o processo de mudança ser totalmente automático e as várias acções dos elementos da arquitectura estarem limitadas a um tempo máximo de resposta. Após a análise dos resultados dos testes do sistema verificou-se que o tempo de resposta a um pedido de FHO é de cerca de cerca de 13 ms. Pensa-se que tão elevado tempo de resposta se deve ao facto de a lista dos routers de sistema ter sido implementada usando uma lista, o que poderá ser melhorado recorrendo a *hash-tables* para o efeito.

Com o objectivo de poder gerir equipamento de rede Cisco foi desenvolvido suporte COPS-RSVP. Os routers são configurados para funcionarem como PEP's de uma rede *IntServ* e a exportarem as decisões para o BB que funcionava como PDP da rede. Os pedidos são feitos através de mensagens COPS onde o BB é contactado no sentido de se pronunciar acerca do pedido de admissão de um novo fluxo, ou da renovação de um fluxo anteriormente aceite. O suporte COPS-RSVP desenvolvido foi para IPv4, uma vez que até à data a Cisco não tem suporte COPS-RSVP

para IPv6. Este suporte foi integrado e testado com sucesso, podendo ainda o BB funcionar em simultâneo como PEP *IntServ* em IPv4 e como PEP *DiffServ* em IPv6.

Foi desenvolvida uma API CLI que permite um login remoto do BB nas máquinas da rede que gere, possibilitando a configuração dos elementos de rede usando CLI.

Foram feitos testes de integração do software desenvolvido na plataforma de testes do projecto Moby Dick existente no Instituto de Telecomunicações e em duas outras do projecto existentes na Universidade Carlos III de Madrid e na Universidade de Estugarda, onde se verificou que foi possível integrar com sucesso os elementos de software desenvolvidos. Foram ainda feitos testes de integração em reuniões do consorcio do projecto, primeiro em Madrid onde se testou pela primeira vez a integração de todo o software, e depois em Estugarda onde se fez a integração final das várias componentes do software desenvolvido. O software foi ainda demostrado publicamente nas conferências IST - Mobile & Wireless Communications Summit 2003 e 4.th Conference on Telecommunications, realizadas em Aveiro em 15-18 de Junho e 18-20 Junho, respectivamente.

Foram feitas ainda algumas medidas do tempo de resposta do BB às solicitações dos outros elementos de rede e testes de consumo de recursos computacionais que constam do capítulo 5.3. Foi verificado que os tempos de resposta do BB obtidos na maioria dos pedidos é da ordem de 4ms, o que é um valor perfeitamente aceitável.

6.1 SUGESTÕES PARA TRABALHO FUTURO

Alguns dos módulos do software desenvolvido podem sofrer melhorias, ou de implementação de novas componentes.

Após a análise dos resultados dos testes de sistema verificou-se que o tempo de resposta a um pedido de FHO pode ser melhorado significativamente se for alterada a implementação da base de dados residente em memória. Actualmente é usado um *vector* de estruturas para armazenar a lista de routers pertencentes à rede do BB. Existem soluções alternativas que podem poupar muito tempo na resposta ao pedido, soluções essas que podem ser por exemplo a utilização de uma *hash-table.*

Com vista a melhorar a performance do BB existem várias alterações que a arquitectura do BB, e consequentemente as arquitecturas dos outros elementos da rede deveriam sofrer:

- Na interface com a RG:
    - adicionar mensagens para fechar canais anteriormente abertos e assim evitar que os canais estejam abertos até que seja atingido o tempo de validade do mesmo;

- adicionar uma mensagem de redimensionamento de um canal, evitando assim que quando o BB pretenda redimensionar largura de banda de um canal tenha que abrir um novo e esperar o anterior feche após o tempo de validade;

- Na interface do AR:
  - alterar a interface por forma a permitir o envio de uma configuração não solicitada de modo a que o BB não tenha que esperar por um pedido de configuração do AR. Actualmente quando o BB pretende fazer uma nova configuração do AR espera por um pedido do AR e envia junto com a resposta uma flag que sinaliza a pretensão de reenviar nova configuração. Quando o AR recebe a resposta com essa flag activa pede então ao BB que lhe envie a nova configuração. O tempo de atraso entre a decisão de alterar a configuração do AR e a sua efectiva alteração deve ser reduzido ao mínimo de modo a se poder optimizar a gestão de rede feita pelo BB;
  - implementar mensagens de configuração parcial do AR. Actualmente o AR recebe a configuração completa sempre que um dos seus parâmetros de configuração necessita de ser alterado. Este processo de configuração pode ser optimizado se forem implementados procedimentos de configuração parcial, enviando-se somente os parâmetros que devem ser alterados na mensagem de configuração;
- Interface com AAAC:
  - alteração da implementação do interface de modo a que a comunicação passe a ser bidireccional entre o AAAC e o BB. Esta bidireccionalidade permitiria ao BB interrogar o servidor de AAAC acerca de um perfil de um utilizador e assim mais facilmente implementar serviços de rede com limite de utilização ou serviços pré-pagos;

O interface CLI do BB poderia ser melhorado por forma a que o BB tivesse retorno de informação das acções de configuração. Para isso dever-se-ia:

- implementar um tempo limite para a resposta a cada comando;
- criar uma listagem de respostas a cada comando, por forma a que o BB pudesse mais facilmente concluir do sucesso das acções de configuração tomadas;

Este tipo de alteração adicionaria muita robustez ao interface e permitiria diminuir os tempos de configuração pelo facto de se conhecer exactamente quando foi executado pelo AR cada comando e quando se pode executar o próximo.

Os algoritmos de controlo de utilização da rede podem também ser melhorados. Só testes exaustivos e com condições de grande sobrecarga dos recursos da rede permitiram definir quais as

alterações que devem ser feitas na implementação. Será de certeza muito importante perceber como se poderá utilizar ganhos estatísticos sem que isso ponha em causa as garantias de QdS que se pretendem fornecer. Outro passo muito importante que deve ser dado com vista a melhorar a eficiência da gestão feita pelo BB é determinar a quantidade dos recursos requisitados pelos MT que será usada de facto. Essa diferença define uma quantidade de recursos que podem ser alocados pelo BB a outros utilizadores melhorando assim a utilização dos recursos da rede.

Finalmente e de acordo com o que é proposto em [21] seria muito interessante dotar o BB de:

- mecanismos de suporte a *broadcast*. O BB podia configurar elementos de rede de modo a que se suportasse *broadcast* podendo assim eliminar a duplicação normal nestas situações. Estes mecanismos serão tanto mais importantes quanto mais se vulgarizar a utilização de redes WI-FI, onde os recursos de rede são escassos.
- melhoramento do interface entre Brokers. Este melhoramento permitiria ao BB conhecer a topologia das redes suas vizinhas permitindo a adaptação do tráfego nas extremidades da sua rede de modo a que as suas características melhor se adaptem à tecnologia da rede para onde se destina.

6.2 SUMÁRIO

Como comentário final pode dizer-se que de acordo com os objectivos propostos, foi implementado um gestor de QdS para redes de acesso heterogéneas. Mais concretamente, foi desenvolvido um gestor de Qualidade de Serviço para uma rede *DiffServ* com mobilidade e com serviços de AAAC. Foi ainda desenvolvido suporte para routers Cisco de uma rede *IntServ* através da utilização de COPS-RSVP.

O desempenho do software foi avaliado através de medição de tempos de resposta e de consumo de recursos da maquina onde corre. Foram também analisadas questões que têm a ver com a escalabilidade da solução, não tendo sido testado pela dificuldade do teste em questão. Os resultados obtidos foram satisfatórios, podendo ainda ser melhorados tanto em termos de performance como em termos de flexibilidade da solução, de acordo com o que se propõe para trabalho futuro.

REFERÊNCIAS

[1] “Serviços de transmissão de dados - Serviço de acesso à Internet - 1º Trimestre de 2003” - Anacom, *http://www.anacom.pt/ template15.jsp?categoryId=1758*, Agosto de 2003.

[2] “Estatísticas de DNS” - Portal FCCN - Fundação para a Computação Científica Nacional, *http://www.fccn.pt/DNS/Estatisticas/ ?in_menu_option=80002*, Agosto de 2003.

[3] "Integrated Services in the Internet: an Overview", Braden, R., et. al., RFC 1633, Julho de 1994.

[4] "Specification of Guaranteed Quality of Service", S. Shenker, et. al., RFC 2212, Setembro de 1997.

[5] "Specification of the Controlled-Load Network Element Service", J. Wroclawski, RFC 2211, Setembro de 1997.

[6] "General Characterization Parameters for Integrated Service Network Elements", S. Shenker, et. al., RFC 2215, Setembro de 1997.

[7] "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", Braden, Ed., et. al., RFC 2205, Setembro de 1997.

[8] “Implementation and Performance simulation of virtualcloack scheduling algorithm in IP Networks”, Nazy Alborz, 1998

[9] "Differentiated Services for the Internet", IETF *DiffServ* Working Group , URL: *http://www.ietf.org/html.charters/DiffServ-charter.html/* (05/07/2003).

[10] “An architecture for Differentiated Services”, S. Blake et al., RFC 2475, Dezembro 1998

[11] "Multiprotocol Label Switching Architecture", E. Rosen, et. al., RFC 3031, Janeiro de 2001.

[12] "Application-Driven Networking: Class of service in IP, Ethernet and ATM Networks", Jonathan Follows, et. al., *http://www.redbooks.ibm.com*, Junho de 2003.

[13] "A Framework for Integrated Services Operation over *DiffServ* Networks", Y. Bernet, et. al., RFC 2998, Novembro de 2000.

[14] "A Translator between Integrated Service/RSVP and Differentiated Service for End-to-End QdS", Eunkyu Lee, et. al., ICT 2003, 10th International Conference on Telecommunications, Volume: 2, 2003.

[15] "A Prototype Implementation for *IntServ* Operation over *DiffServ* Networks", Werner Almesberger, et. al., IEEE Globecom 2000, S. Francisco, Dezembro de 2000.

[16] “A Survey on Policy-Based Networking – Final Report”, INTAP, 2001

[17] “Terminology for Policy-Based Management”, A. Westerinen et Al., RFC 3198, Novembro de 2001

[18] "Use of COPS for *IntServ* operations over *DiffServ*: Architectural issues, Protocol design and Test-bed implementation", Roberto Mameli, et. al., ICC 2001, Junho de 2001.

[19] "Supporting RSVP in a Differentiated Service Domain: an Architectural Framework and a Scalability Analysis", Andrea Detti, et. al., MQOS workshop, Janeiro de 2001.

[20] "A concept for RSVP over *DiffServ*”, R. Balmer, et. al., Ninth International Conference on Computer Communications and Networks, Abril de 2000.

[21] "QdS Control support for heterogeneous networks", Pedro Gonçalves, et. al., CRC2003, Setembro de 2003.

[22] "Integrated Service Mappings on IEEE 802 Networks", Yavatkar, et al., RFC 2815, Maio de 2000.

[23] "Subnet Bandwidth Manager: A Protocol for RSVP-based Admission Control over IEEE 802-style networks", R. Yavatkar, et. al., RFC 2814, Maio de 2000.

[24] "A Framework for Integrated Services Over Shared and Switched IEEE 802 LAN Technologies", A. Ghanwani, et. al., RFC 2816, Maio de 2000.

[25] “Remote Authentication Dial In User Service (RADIUS)”, C. Rigney, et. al., RFC 2138, Abril de 1997.

[26] "A Framework for Policy-based Admission Control", Yavatkar, et al., RFC 2753, Janeiro de 2000.
[27] "Signaled Preemption Priority Policy Element", S. Herzog, RFC 2751, Janeiro de 2000.
[28] "Generic AAA Architecture", C. de Laat , et. al.e, RFC 2903, Agosto de 2000.
[29] ”Criteria for Evaluation AAA Protocols for Network Access”, B. Aboba, et. al., RFC 2989, Novembro de 2000.
[30] ”Diameter Base Protocol”, Pat R. Calhoun, et. al., draft-ietf-aaa-diameter-08-alpha01.txt, IETF work in progress, August 2001, URL: *http://www.ietf.org/internet-drafts/draft-ietfaaa-diameter-08.txt*. (05/07/2003).
[31] ”Diameter CMS Security Application”, Pat R. Calhoun, et. al., draft-ietf-aaa-diameter-cms-03.alpha01.txt, IETF work in progress, August 2001, URL:*http://www.ietf.org/internet-drafts/draft-ietf-aaa-diameter-cmssec-03.txt*.(05/07/2003).
[32] ”Diameter Mobile IPv4 Application”, Pat R. Calhoun, et. al., draftietf-aaa-diameter-mobileip-08.txt, IETF work in progress, November 2001, URL: *http://www.ietf.org/internet-drafts/draft-ietf-aaa-diameter-mobileip-09.txt*.(05/07/2003).
[33] ”Diameter NASREQ Application”, Pat R. Calhoun, et. al., draft-ietf-aaa-diameter-nasreq-08-alpha01.txt, IETF work in progress, August 2001, URL: *http://www.ietf.org/internet-drafts/draft-ietfaaa-diameter-nasreq- 08.txt*. (05/07/2003).
[34] "The COPS (Common Open Policy Service) Protocol", Durham, et al., RFC 2748, Janeiro de 2000.
[35] "HMAC: Keyed-Hashing for Message Authentication", Krawczyk, H., et. al., RFC 2104, Fevereiro de 1997.
[36] "The MD5 Message-Digest Algorithm", Rivest, R., RFC 1321, Abril de 1992.
[37] "COPS usage for RSVP", Herzog, et al., Janeiro de 2000.
[38] "COPS Usage for Policy Provisioning (COPS-PR)", Chan, et al., RFC 3084, Março de 2001
[39] "A Framework for Policy Based Admission Control", Yavatkar, et. al., RFC 2753, Janeiro de 2000.
[40] "An Engineering Approach to Computer Networking", S. Keshav, 1977, Addison-Wesley.
[41] “A Discussion of Bandwidth Broker Requirements for Internet2 Qbone Deployment”, Rob Neilson, et. al., Version 0.7, Janeiro de 2004.
[42] “The Siemens Bandwith Broker” *http://qos.internet2.edu/houston2000/proceedings/Stelzl/20000209-QoS2000-Stelzl.pdf*”, Janeiro de 2004.
[43] “Implementation of the Two Tier Differentiated Services Architecture”, UCLA-IRL, (*http://irl.cs.ucla.edu/twotier/*), Janeiro de 2004.
[44] “Preparing the Network Infrastructure for the Information Economy in Europe”, *www.ssgrr.it/en/ssgrr2002w/papers/46.pdf*, P. Polese et Al., Janeiro de 2004
[45] Differentiated Services – implementation, (*http://www.ittc.ukans.edu /~kdrao/BB/*), D. Dhananjava, Agosto de 1999.
[46] “QBone Bandwidth Broker Architecture”, *http://qbone.internet2.edu/bb/bboutline2.html*.
[47] The apache software Foundation , (*http://www.apache.org/*), Setembro de 2003-09-05.
[48] MySQL (*http://www.mysql.com/*), Setembro de 2001.
[49] “Simple Network Management Protocol (SNMP)”, RFC 1098, J.D. Case, et. al., Abril de 1989.
[50] Simple Inter Bandwidth Broker Singnalling, *http://qos.internet2.edu/wg/documents-informational/20020709-chimento-etal-qbone-signaling/SIBBS-SEC.html*, Setembro de 2003.
[51] Projecto Moby Dick, *http://ist-Moby Dick.org*, Setembro de 2003.
[52] “Design and Evaluation of a Handover Decision Strategy for 4th Generation Mobile Networks”, Wenhui Zhang et Al., The 57th IEEE Semiannual Vehicular Technology Conference, Jeju, Korea, Abril de 2003.
[53] *http://kcachegrind.sourceforge.net/cgi-bin/show.cgi/KcacheGrindWhat*, Agosto de 2003.

GLOSSÁRIO

Rede de Acesso (Access Network)	Rede ligada as restantes por um Access Router, que permite a ligação de um ou mais Nós IP.
Router de acesso (Access Router)	Um router que resida na periferia de uma rede de acesso e que esteja ligado a um ou mais pontos de acesso. O router de acesso pode incluir outras funcionalidades não disponíveis nos routers IP normais.
Autenticação (*Authentication*)	O acto de verificar uma identidade anunciada, na forma de uma etiqueta preexistente mutuamente conhecida tanto do originador da mensagem como destino (entidade de autenticação)
Autorização (*Authorization*)	Uma autorização é um direito ou uma permissão que é garantida a uma entidade do sistema de modo a aceder a um recurso do sistema. Um processo de autorização é um procedimento com vista ao fornecimento de direitos.
Care-of Address	Ponto de terminação do túnel no MN, para datagramas encaminhados até ao MN, quando este está for a do seu *Home Domain*.
Correspondent Node	Um parceiro com o qual o MN está a comunicar. Um CN pode ser tanto móvel com estático.
Encriptação (*Encryption*)	Transformação dos dados (chamados *plaintext*) para uma forma (chamada *ciphertext*) que contém a informação original, com o objectivo de prevenir que seja conhecida ou usada. Se a transformação for reversível o processo inverso chama-se desencriptação e devolve os dados encriptados ao seu estado original.
Foreign domain	Um domínio administrativo, visitado por um cliente com suporte *Mobile IP*, e que contenha a infra-estrutura AAA necessária para executar as operações necessárias para permitir registos *Mobile IP*.
Handover ou handoff	*Handover* é o processo que tem lugar quando um MH no estado activo muda de pondo de ligação à rede, ou quando essa mudança é tentada. A rede de acesso pode fornecer funcionalidades especiais para minimizar a interrupção para as sessões activas durante o processo.
Home Address	Um endereço IP que está atribuído por um largo período de tempo a um MN. O endereço não é alterado independentemente do ponto onde o nó está ligado à rede.
Home domain	Um domínio administrativo que contenha infra-estrutura de AAA de interesse relevante para um cliente Mobile IP que esteja numa rede estrangeira.
Home Network	Uma rede, possivelmente virtual, que tendo um prefixo de rede que

corresponda com o prefixo de rede do *home address* do MN. É de realçar que os mecanismos de encaminhamento IP irão entregar os datagramas destinados ao *home address* do MT à *Home Network* do MN.